R:BASE 11

for Windows

Reference Index

R:BASE 11 for Windows

Reference Index

by R:BASE Technologies, Inc.

Welcome to R:BASE 11 for Windows!

R:BASE 11 is the new major release from R:BASE Technologies showcasing the latest advances for your databases and applications.

R:BASE 11 combines pure performance and high-tech features with a practical interface for users of all levels. R:BASE 11 users can achieve peak productivity to overachieve, outperform, and over-deliver on data management goals.

R:BASE 11 Reference Index

Copyright © 1982-2024 R:BASE Technologies, Inc.

Information in this document, including URL and other Internet web site references, is subject to change without notice. The example companies, individuals, products, organizations and events depicted herein are completely fictitious. Any similarity to a company, individual, product, organization or event is completely unintentional. R:BASE Technologies, Inc. shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance, or use of this material. This document contains proprietary information, which is protected by copyright. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written consent of R:BASE Technologies, Inc. We reserve the right to make changes from time to time in the contents hereof without obligation to notify any person of such revision or changes. We also reserve the right to change the specification without notice and may therefore not coincide with the contents of this document. The manufacturer assumes no responsibilities with regard to the performance or use of third party products.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of that agreement. Any unauthorized use or duplication of the software is forbidden.

R:BASE Technologies, Inc. may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from R:BASE Technologies, Inc., the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Trademarks

R:BASE®, Oterro®, RBAdmin®, R:Scope®, R:Mail®, R:Charts®, R:Spell Checker®, R:Docs®, R:BASE Editor®, R:BASE Plugin Power Pack®, R:Style®, RBZip®, R:Mail Editor®, R:BASE Dependency Viewer®, R:Archive®, R:Chat®, R:PDF Form Filler®, R:FTPClient®, R:SFTPClient®, R:PDFWorks®, R:Magellan®, R:WEB Reports®, R:WEB Gateway®, R:PDFMerge®, R:PDFSearch®, R:Documenter®, RBInstaller®, RBUpdater®, R:AmazonS3®, R:GAP®, R:Mail Viewer®, R:Capture®, R:Synchronizer®, R:Biometric®, R:CAD Viewer®, R:DXF®, R:Twain2PDF®, R:Tango®, R:Scheduler®, R:Scribbler®, R:SmartSig®, R:OutLink®, R:HASH®, R:JobTrack®, R:TimeTrack®, R:Manufacturing®, R:GeoCoder®, R:Code®, R:Fax®, R:QBDataDirect®, R:QBSynchronizer®, R:QBDBExtractor®, and Pocket R:BASE® are trademarks or registered trademarks of R:BASE Technologies, Inc. All Rights Reserved. All other brand, product names, company names and logos are trademarks or registered trademarks of their respective companies.

Windows, Windows 11-10, Windows Server 2022-2012, Bing Maps, Word, Excel, Access, SQL Server, and Outlook are registered trademarks of Microsoft Corporation. OpenOffice is a registered trademark of the Apache Software Foundation.

Printed: April 2024 in Murrysville, PA

First Edition

Table of Contents

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Part I

1 Reference Index

1.1 Aggregate Functions

An aggregate function can be used to provide summary data for a row in a table or for a provided list of values.

Function	Supported Areas	Description
AVG	SELECT, COMPUTE, LAVG, Data Browser, Query Wizard, Query Builder, Form/Report/Label Expressions	Computes the numeric average. R:BASE rounds averages of integer values to the nearest integer value and currency values to their nearest unit.
COUNT	SELECT, COMPUTE, Data Browser, Query Wizard, Query Builder, Form/Report/Label Expressions	Determines how many non-null entries there are for a particular column item.
LISTOF	SELECT, Query Builder, Form/Report/Label Expressions	Creates a text string of the values separated by the current comma delimiter character. The LISTOF function can be used to populate a variable with a list of values from multiple rows.
MAX	SELECT, COMPUTE, LMAX, Data Browser, Query Wizard, Query Builder, Form/Report/Label Expressions	Selects the maximum numeric, time, date, or alphabetic value in a column.
MIN	SELECT, COMPUTE, LMIN, Data Browser, Query Wizard, Query Builder, Form/Report/Label Expressions	Selects the minimum numeric, time, date, or alphabetic value in a column.
STDEV	SELECT, COMPUTE, Data Browser, Query Builder	Computes standard deviation. The standard deviation is a measure of how widely values are dispersed from the average value.
SUM	SELECT, COMPUTE, LAVG, Data Browser, Query Wizard, Query Builder, Form/Report/Label Expressions	Computes the numeric sum.
VARIANCE	SELECT, COMPUTE, Data Browser, Query Builder	Determines variance.

***** Selecting aggregate functions, such as MIN and MAX, requires that R:BASE keeps an accumulator and choose to only use the **first 80 characters** for NOTE values. This matches the fact that if you sort on NOTE fields, the sort will be based on the first 80 characters only.

1.2 Balloon Tip Hints

The balloon tip is a hint-type of notification that is used to display a message to an end user running a form. The balloon tip is usually displayed as a response to an action made by the user. The balloon tip can be used with DB Edit and Variable Edit controls, and the settings are available on the "Additional" tab of the object properties. A title, message, and icon image can be displayed in the balloon tip.

⇨ **Balloon Tip**

Title
Specifies the tip title. The maximum length for the balloon tip title is 99 characters.

Text
Specifies the tip text

Icon
Specifies the icon displayed in the tip. Icon options available include:

Copyright © 1982-2024 R:BASE Technologies, Inc.

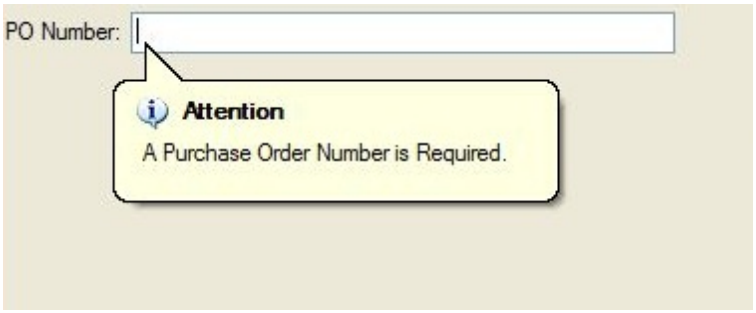
• None • Info • Warning • Error • Info Large * • Warning Large * • Error Large *	* (Windows Vista, 7, and higher only)

The balloon tip will only appear when the SHOWBALLOONTIP parameter is set to TRUE for the DB Edit or Variable Edit control's Component ID.

Sample:

```
PROPERTY PEdit_PO BalloonTipTitle 'Attention'
PROPERTY PEdit_PO BalloonTipText 'A Purchase Order Number is Required.'
PROPERTY PEdit_PO BalloonTipIcon Info

IF cPurchaseOrder IS NULL THEN
  PROPERTY PEdit_PO ShowBalloonTip TRUE
ELSE
  PROPERTY PEdit_PO ShowBalloonTip FALSE
ENDIF
```



1.3 Binary Large Objects (BLOB)

Binary Large Objects or BLOBs refer to images that you can store within your R:BASE database. You can add, edit, or delete Binary Large Objects (BLOBs) within your database files. The recommended table data type to store images is VARBIT. The R:BASE BLOB Editor has also been enhanced to manage Multipage Images. This enhancement will allow users to manage multipage images, such as .DCX, .GIF, or .TIFF files, when saved as BLOB data in R:BASE.

The R:BASE BLOB Editor also works with your large ASCII data files (Large Objects or LOBs). These objects refer to text files that can be in any ASCII format, including RTF. The recommended table data type for large text files is VARCHAR.

The data for VARBIT and VARCHAR data types is stored in the fourth R:BASE database file.

See also:

Data Types
RBBEDIT
R:BASE BLOB Editor

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.3.1 Loading BLOB/LOB Data

You can easily load VARBIT and/or VARCHAR data into R:BASE tables directly from the Data Browser.

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

See also:

Data Types
RBBEDIT
R:BASE BLOB Editor

1.3.2 Using Commands with BLOBs

Use the INSERT or LOAD commands to add binary large objects to your database. After the binary large object is loaded, the file is in the database, so you do not need the disk file. Following is an example of an INSERT command that adds a binary large object to a database:

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

Binary large objects can be placed for viewing in forms and reports by placing the column or variable containing the binary large object in the form or report. When you run a form, you can enter or edit a reference to a binary large object by pressing [Shift] + [F10]. In an application program, you can use the SET VARIABLE command to define a variable that is equal to the file name that contains the binary large object.

You can write a variable that has a VARBIT or VARCHAR data type back to a file using the WRITE command, for example:

```
WRITE .v1 TO filename
```

The BACKUP and UNLOAD commands create a file with a .LOB extension for binary large objects, and a file for the data and/or structure.

See also:

Data Types
RBBEDIT
R:BASE BLOB Editor
SELECT

1.4 Building Advanced Queries

1.4.1 Building a Query Using a Union

A "Union" can be added to your queries in order to combine the results of two or more SELECT statements. This optional operator combines the results of two SELECT commands or clauses, displaying the results of the second SELECT command below those of the first. By default, unions deletes duplicate rows.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Include the optional keyword ALL to include duplicate rows in the final result.

The UNION operator requires the following three conditions:

- The SELECT statements must specify an equal number of columns.
- Columns that are being combined must have the same data type.
- Only the last SELECT statement can contain an ORDER BY clause.

1.4.1.1 Quarterly Summary Query

In the RRBYW20 sample database, the "QuarterlySummary" view example utilizes three UNION statements with an initial SELECT statement. The following instructions will step through the process to recreate the view in the Query Builder.

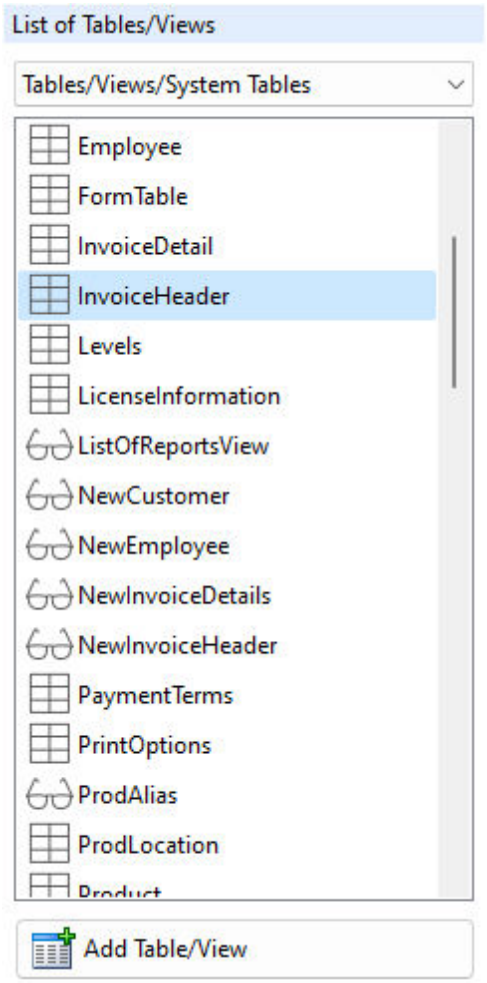
To continue, launch R:BASE 11 and connect to the RRBYW20 sample database. The database is located in the following default installation directory: "C:\RBTI\RBG11\Samples\RRBYW20".

1. Start R:BASE
2. Connect to the RRBYW20 sample database, by selecting "Database" > "Connect", and navigating to the above default installation directory based upon your version.
3. Then, launch the Query Builder by selecting "Tools" > "Query by Example" from the main Menu Bar

1.4.1.1.1 1st Quarter

In the Query Builder you will see a list of tables and views within a panel to the left that can be added to your query. The list contains all tables and views for the connected database. The table used in this example is InvoiceHeader.

Copyright © 1982-2024 R:BASE Technologies, Inc.



1. Add the "InvoiceHeader" to the query by selecting the table, and then selecting the "Add Table/View" button.

   The table should now be listed under "Tables/Views In Use". With the table added, the specific columns to be used in the query must be selected.

2. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "Select Columns" from the speed menu.

   The "InvoiceHeader" columns will be displayed to be added to the query. This dialog window is basically a graphic representation of the SELECT command. When working with columns in your queries, expressions can also be added to the columns to perform calculations.

3. In this example, select the "New Expression" button to display the Expression Builder.
4. Enter the value "1" into the "Expression Text" panel. The value "1" is the numeric value that will represent the 1st quarter of the figures in the query. Press the "OK" button to save the expression.
5. Select the "New Expression" button to once again display the Expression Builder.
6. Within the the "Function Templates" panel, select use the drop-down combo box to select the aggregate function "SUM", which will be displayed as "(SUM(arg))"
7. Select the "Use" button to add the function to the "Expression Text" panel.
8. Place the cursor within the set of parenthesis and delete the "arg" value.
9. Then, with the cursor still in the parenthesis, use the "Select Columns" list box to locate and select the "InvoiceTotal" column.
10. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.

    The Expression panel should have the following displayed:

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
(SUM( T1.InvoiceTotal ))
```

11. Click the OK button to close the Expression Builder.
12. Click the OK button to close the column selection dialog and return to the Query Builder main window.
13. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "WHERE Clause" from the speed menu.

The WHERE Builder will be displayed to place a condition on the view in order to limit the records displayed.

14. Use the "Select Columns" list box to locate and select the "TransDate" column.
15. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.
16. From the "Operators and Keywords" panel, select the "BETWEEN" button to add the keyword to the WHERE Clause panel.
17. Type in the **first** quarter starting date of "01/01/2018".
18. From the "Operators and Keywords" panel, select the "AND" button to add the keyword to the WHERE Clause panel.
19. Type in the **first** quarter ending date of 03/31/2018".

The Expression panel should have the following displayed:

```
TransDate BETWEEN 01/01/2018 AND 03/31/2018
```

20. Click the OK button to close the WHERE Builder.

Back in the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT 1,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE TransDate BETWEEN 01/01/2018 AND 03/31/2018
```

1.4.1.1.2 2nd Quarter

To add the 2nd Quarter results, we will begin by adding a UNION to the query. A UNION can be added to queries in order to combine the results of two or more SELECT statements.

1. Select the "Add UNION" button from Query Builder Toolbar.

Take notice that a "UNION 1" tab has appeared in the Query Builder. These tabs permit switching from multiple UNIONs within a query.

After this point, the steps are very similar to building the 1st quarter statement.

2. Add the "InvoiceHeader" to the query by selecting the table, and then selecting the "Add Table/View" button.

The table should now be listed under "Tables/Views In Use". With the table added, the specific columns to be used in the query must be selected.

3. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "Select Columns" from the speed menu.
4. In this example, select the "New Expression" button to display the Expression Builder.
5. Enter the value "2" into the "Expression Text" panel. The value "2" is the numeric value that will represent the 2nd quarter of the figures in the query. Press the "OK" button to save the expression.
6. Select the "New Expression" button to once again display the Expression Builder.
7. Within the the "Function Templates" panel, select use the drop-down combo box to select the aggregate function "SUM", which will be displayed as "(SUM(arg))"
8. Select the "Use" button to add the function to the "Expression Text" panel.
9. Place the cursor within the set of parenthesis and delete the "arg" value.

Copyright © 1982-2024 R:BASE Technologies, Inc.

10. Then, with the cursor still in the parenthesis, use the "Select Columns" list box to locate and select the "InvoiceTotal" column.
11. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.

    The Expression panel should have the following displayed:

    ```
    (SUM( T1.InvoiceTotal ))
    ```

12. Click the OK button to close the Expression Builder.
13. Click the OK button to close the column selection dialog and return to the Query Builder main window.
14. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "WHERE Clause" from the speed menu.
15. Use the "Select Columns" list box to locate and select the "TransDate" column.
16. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.
17. From the "Operators and Keywords" panel, select the "BETWEEN" button to add the keyword to the WHERE Clause panel.
18. Type in the **second** quarter starting date of "04/01/2018".
19. From the "Operators and Keywords" panel, select the "AND" button to add the keyword to the WHERE Clause panel.
20. Type in the **second** quarter ending date of 06/30/2018".

    The Expression panel should have the following displayed:

    ```
    TransDate BETWEEN 04/01/2018 AND 06/30/2018
    ```

21. Click the OK button to close the WHERE Builder.

    In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

    ```
    SELECT 1,(SUM( T1.InvoiceTotal ))
    FROM  InvoiceHeader T1
    WHERE T1.TransDate BETWEEN 01/01/2018 AND 03/31/2018
    UNION SELECT 2,(SUM( T1.InvoiceTotal ))
    FROM  InvoiceHeader T1
    WHERE T1.TransDate BETWEEN 04/01/2018 AND 06/30/2018
    ```

1.4.1.1.3 3rd Quarter

To add the 3rd Quarter results, we will add another UNION to the query.

1. Select the "Add UNION" button from Query Builder Toolbar.

   Take notice that a "UNION 2" tab has appeared in the Query Builder.

2. Add the "InvoiceHeader" to the query by selecting the table, and then selecting the "Add Table/View" button.
3. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "Select Columns" from the speed menu.
4. In this example, select the "New Expression" button to display the Expression Builder.
5. Enter the value "3" into the "Expression Text" panel. The value "3" is the numeric value that will represent the 3rd quarter of the figures in the query. Press the "OK" button to save the expression.
6. Select the "New Expression" button to once again display the Expression Builder.
7. Within the the "Function Templates" panel, select use the drop-down combo box to select the aggregate function "SUM", which will be displayed as "(SUM(arg))"
8. Select the "Use" button to add the function to the "Expression Text" panel.
9. Place the cursor within the set of parenthesis and delete the "arg" value.
10. Then, with the cursor still in the parenthesis, use the "Select Columns" list box to locate and select the "InvoiceTotal" column.
11. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.
12. Click the OK button to close the Expression Builder.

Copyright © 1982-2024 R:BASE Technologies, Inc.

13. Click the OK button to close the column selection dialog and return to the Query Builder main window.
14. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "WHERE Clause" from the speed menu.
15. Use the "Select Columns" list box to locate and select the "TransDate" column.
16. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.
17. From the "Operators and Keywords" panel, select the "BETWEEN" button to add the keyword to the WHERE Clause panel.
18. Type in the **third** quarter starting date of "07/01/2018".
19. From the "Operators and Keywords" panel, select the "AND" button to add the keyword to the WHERE Clause panel.
20. Type in the **third** quarter ending date of 09/30/2018".

    The Expression panel should have the following displayed:

    ```
    TransDate BETWEEN 07/01/2018 AND 09/30/2018
    ```

21. Click the OK button to close the WHERE Builder.

    In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

    ```
    SELECT 1,(SUM( T1.InvoiceTotal ))
    FROM  InvoiceHeader T1
    WHERE TransDate BETWEEN 01/01/2018 AND 03/31/2018
    UNION SELECT 2,(SUM( T1.InvoiceTotal ))
    FROM  InvoiceHeader T1
    WHERE T1.TransDate BETWEEN 04/01/2018 AND 06/30/2018
    UNION SELECT 3,(SUM( T1.InvoiceTotal ))
    FROM  InvoiceHeader T1
    WHERE T1.TransDate BETWEEN 07/01/2018 AND 09/30/2018
    ```

1.4.1.1.4 4th Quarter

To add the 4th Quarter results, we will add a final UNION to the query.

1. Select the "Add UNION" button from Query Builder Toolbar.
2. Add the "InvoiceHeader" to the query by selecting the table, and then selecting the "Add Table/View" button.
3. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "Select Columns" from the speed menu.
4. In this example, select the "New Expression" button to display the Expression Builder.
5. Enter the value "4" into the "Expression Text" panel. The value "4" is the numeric value that will represent the 4th quarter of the figures in the query. Press the "OK" button to save the expression.
6. Select the "New Expression" button to once again display the Expression Builder.
7. Within the the "Function Templates" panel, select use the drop-down combo box to select the aggregate function "SUM", which will be displayed as "(SUM(arg))"
8. Select the "Use" button to add the function to the "Expression Text" panel.
9. Place the cursor within the set of parenthesis and delete the "arg" value.
10. Then, with the cursor still in the parenthesis, use the "Select Columns" list box to locate and select the "InvoiceTotal" column.
11. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.
12. Click the OK button to close the Expression Builder.
13. Click the OK button to close the column selection dialog and return to the Query Builder main window.
14. Right click on the "InvoiceHeader" table under "Tables/Views In Use", and choose "WHERE Clause" from the speed menu.
15. Use the "Select Columns" list box to locate and select the "TransDate" column.
16. Once selected, press the "Add Column" button to insert the column into the "Expression Text" panel.
17. From the "Operators and Keywords" panel, select the "BETWEEN" button to add the keyword to the WHERE Clause panel.
18. Type in the **fourth** quarter starting date of "10/01/2018".
19. From the "Operators and Keywords" panel, select the "AND" button to add the keyword to the WHERE Clause panel.
20. Type in the **fourth** quarter ending date of 12/31/2018".

Copyright © 1982-2024 R:BASE Technologies, Inc.

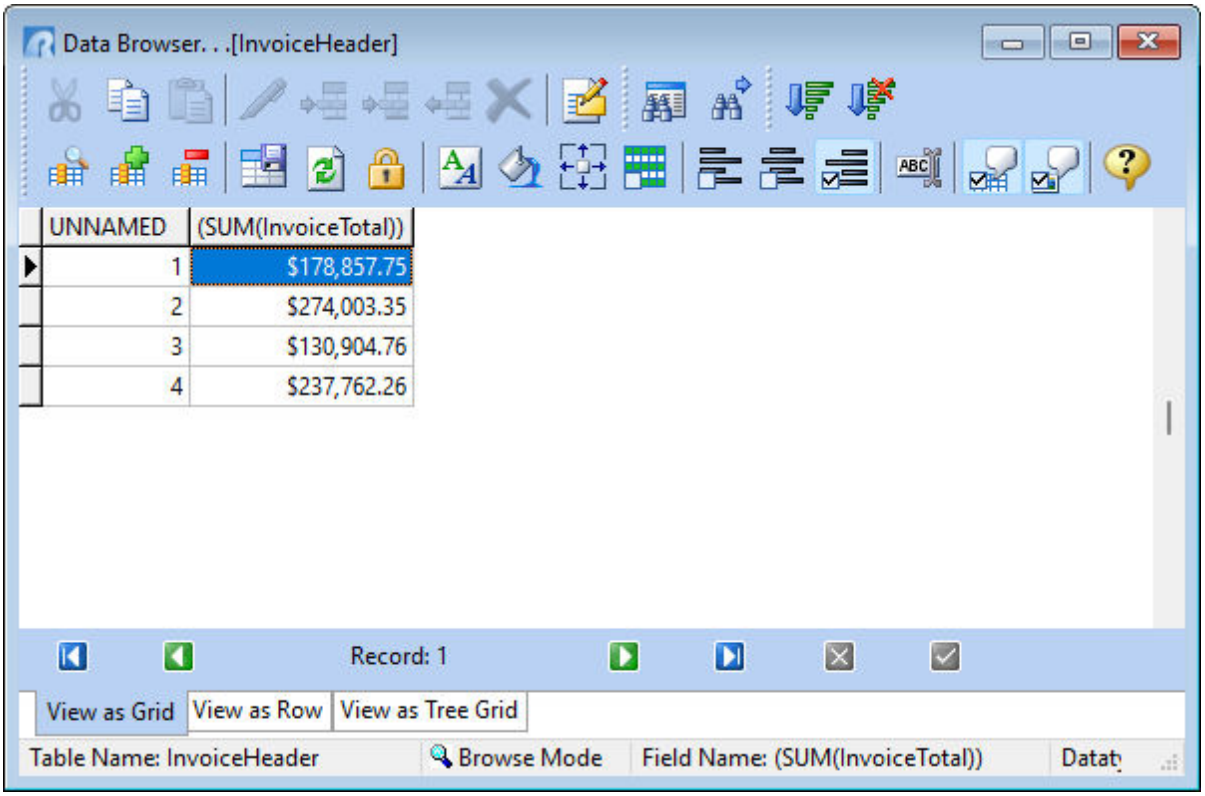
21. Click the OK button to close the WHERE Builder.

In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT 1,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE TransDate BETWEEN 01/01/2018 AND 03/31/2018
UNION SELECT 2,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE T1.TransDate  BETWEEN 04/01/2018 AND 06/30/2018
UNION SELECT 3,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE T1.TransDate BETWEEN 07/01/2018 AND 09/30/2018
UNION SELECT 4,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE T1.TransDate BETWEEN 10/01/2018 AND 12/31/2018
```

1.4.1.1.5 Browse the Query Results

At this point the query results can be reviewed by selecting the "Browse Query" button, which will display the data in the Data Browser.



1.4.1.1.6 Column Aliases

With non-descriptive column headings, the Query Builder allow the ability to assign column aliases. By using column aliases, when the query is displayed, the results will be easier to understand.

1. If you have not done so already, close the Data Browser by pressing the [Esc] key.
2. From the main Menu Bar, choose "Query" > "Column Aliases.."

The "Create Alias" dialog will appear.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3. First, enable aliases by selecting the "Use Column Aliases" check box.

   The panel will become editable with the "Column Name", "Alias Name", and "Modify" panels.

4. Select the first item in the "Column Name" panel.
5. The corresponding value displayed in the "Alias Name" will then appear in the "Modify" panel below.
6. Overwrite the value with the word "Quarter", and select the "Modify Current Selection" button.
7. Select the first item in the "Column Name" panel.
8. The corresponding value displayed in the "Alias Name" will then appear in the "Modify" panel below.
9. Overwrite the value with the word "TotalSales", and select the "Modify Current Selection" button.
10. Press the OK button to save the aliases.

    In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
(Quarter,TotalSales)
SELECT 1,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE TransDate BETWEEN 01/01/2018 AND 03/31/2018
UNION SELECT 2,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE T1.TransDate  BETWEEN 04/01/2018 AND 06/30/2018
UNION SELECT 3,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE T1.TransDate BETWEEN 07/01/2018 AND 09/30/2018
UNION SELECT 4,(SUM( T1.InvoiceTotal ))
FROM  InvoiceHeader T1
WHERE T1.TransDate BETWEEN 10/01/2018 AND 12/31/2018
```

1.4.1.1.7 Save the Query as a View

To display the alias values, first save the query.

1. From the main Menu Bar, select "File" > "Save Query As View..."
2. In the dialog, enter "QuarterlySales2018" within the "View Name:" field.
3. In the dialog, enter "2018 Quarterly Sales Figures" within the "View Comment:" field.
4. Select the "OK" button.
5. Close the Query Builder by selecting "File" > "Close" from the Menu Bar.

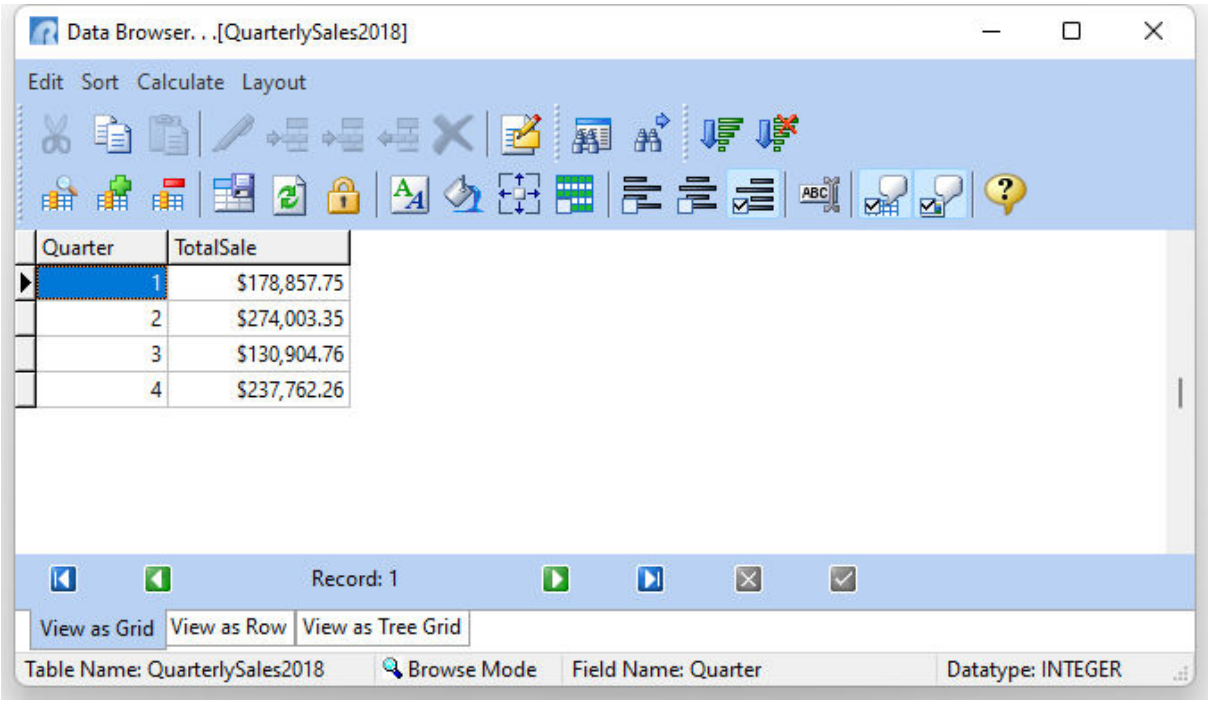
   In the Database Explorer, the view will be displayed with the name, comment, and number of columns

1.4.1.1.8 Browse the View

To launch the view, double click on "QuarterlySales2018"

The results should appear like the following.

Copyright © 1982-2024 R:BASE Technologies, Inc.



1.4.2 Building Queries Using Joins

A "Join" is an SQL clause that combines records from two tables in a database. When you perform a join, you specify one column from each table to join on. These two columns contain data that is shared across both tables. You can use multiple joins in the same SQL statement to query data from as many tables as you like. Join types include Inner, Left Outer, Right Outer, and Full Outer.

The differences are:

- **Inner Join** - returns rows when there is at least one row in both tables that match the join condition
- **Left Outer Join** - returns rows that have data in the left table (left of the JOIN keyword), even if there's no matching rows in the right table
- **Right Outer Join** - returns rows that have data in the right table (right of the JOIN keyword), even if there's no matching rows in the left table
- **Full Outer Join:** - returns all rows, as long as there's matching data in one of the tables

Most of the time, you'll do an Inner Join, though you will sometimes find it useful to do an Outer Join. For example, you need an Outer Join to get all rows in the following cases:

- When joining a *customer* table with an *orders* table to list the customers who ordered something in the current month as well as those who didn't order anything.
- When joining a *budget* table with an *expense* table to list each budget item, whether or not there was an expense for that item in the current month.
- When comparing a header (master table) on the "one" side of a one-to-many relationship against a detail (transaction table) on the "many" side to see all the rows of data, whether or not they have associated details.

1.4.2.1 Inner Join

When using an Inner Join, rows are returned when there is at least one row in both tables that match the join condition. Inner Joins are the most common join operation and is the default for R:BASE joins.

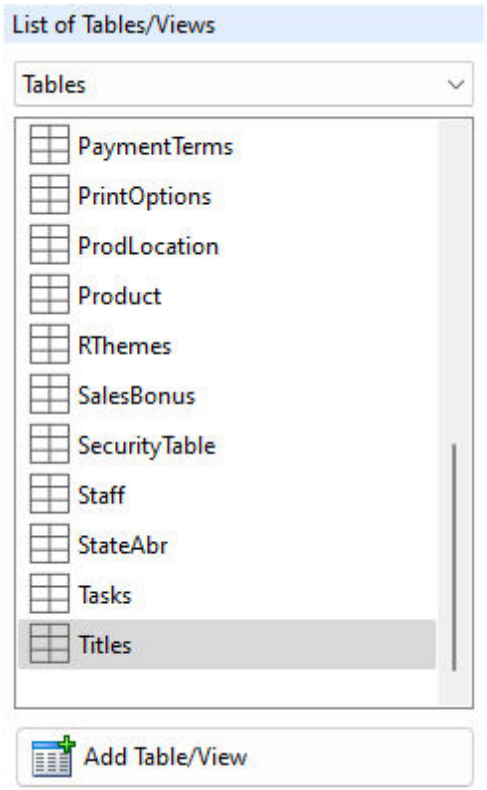
The following instructions will step through the process to recreate an Inner Join in the Query Builder.

Copyright © 1982-2024 R:BASE Technologies, Inc.

To continue, launch R:BASE 11 and connect to the RRBYW20 sample database. The database is located in the following default installation directory: "C:\RBTI\RBG11\Samples\RRBYW20".

1. Start R:BASE
2. Connect to the RRBYW20 sample database, by selecting "Database" > "Connect", and navigating to the above default installation directory based upon your version.
3. Then, launch the Query Builder by selecting "Tools" > "Query by Example" from the main Menu Bar

In the Query Builder you will see a list of tables and views within a panel to the left that can be added to your query. The list contains all tables and views for the connected database.



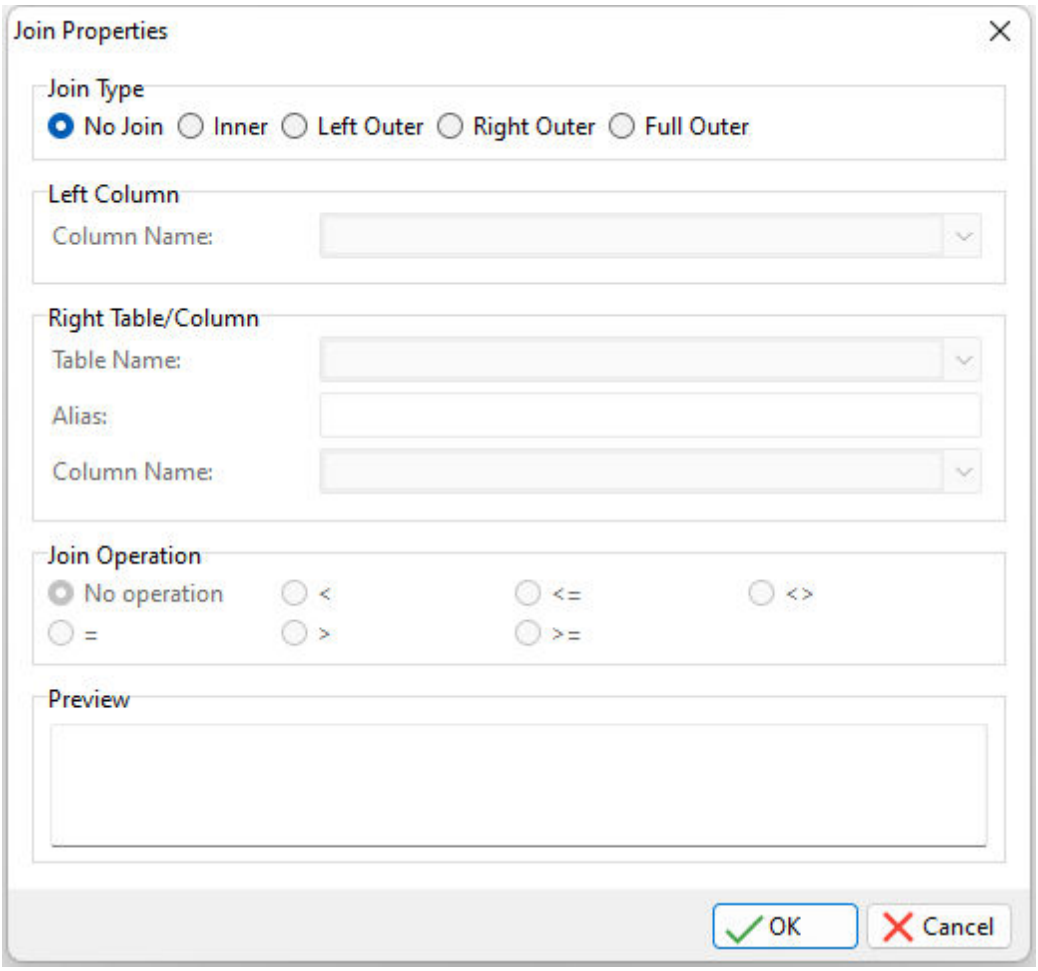
4. Add the "Titles" table to the query by selecting the table, and then selecting the "Add Table/View" button.

The table should now be listed under "Tables/Views In Use".

5. Right click on the "Titles" table under "Tables/Views In Use", and choose "Join Properties..." from the speed menu.

The "Join Properties" dialog will be displayed.

Copyright © 1982-2024 R:BASE Technologies, Inc.



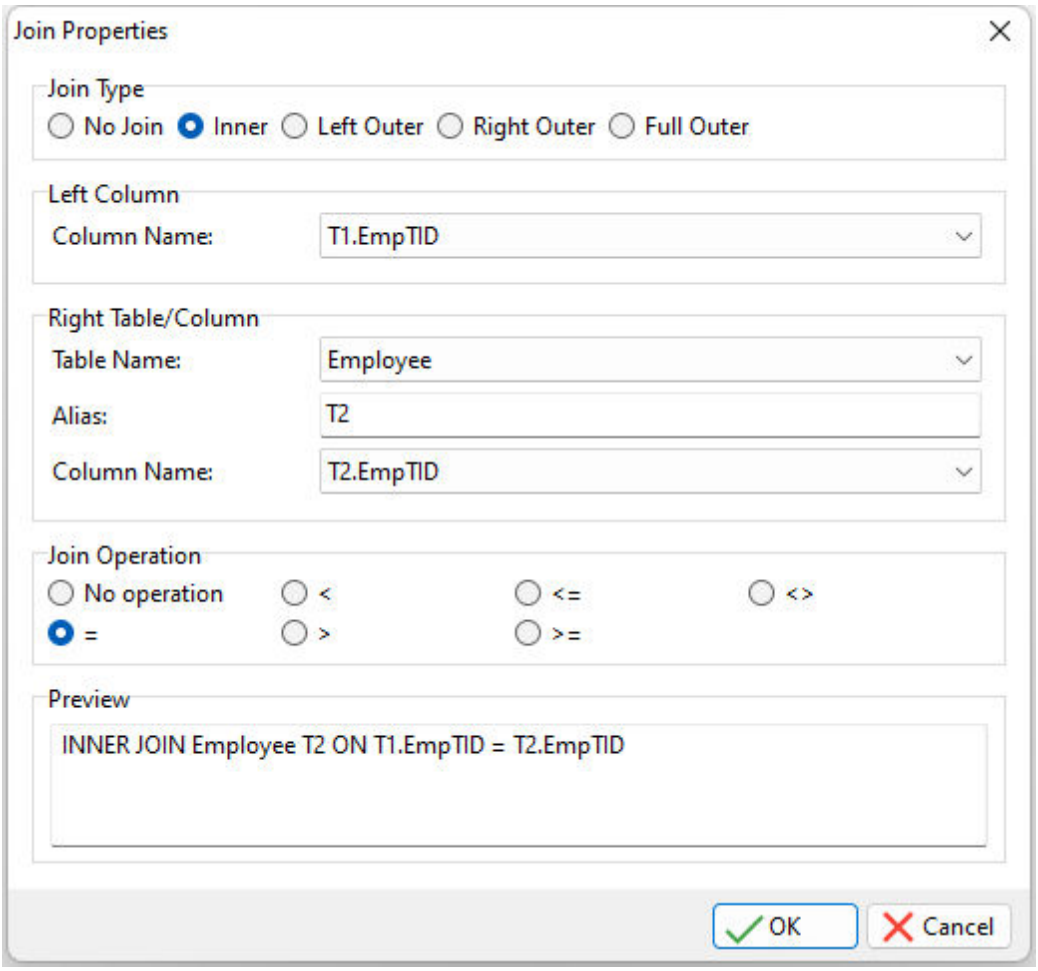
6. From the "Join Type" radio button options, select "Inner"

   Take note that the other options within the window will become enabled. In this window you will select the linking columns for the join, the second table for the join, and the join operation. As alias can also be assigned to the linking column.

7. From the "Left Column" panel, choose "T1.EmpTID" from the "Column Name:" drop down box. The "T1" alias may vary from your R:BASE screen.
8. From the "Right Table/Column" panel, choose "Employee" from the "Table Name:" drop down box.
9. From the "Right Table/Column" panel, enter "T2" into the "Alias:" field to assign the table alias for the join. If "T2" is already used for the "Titles" table, then use "T3" as the alias.
10. Again in the "Right Table/Column" panel, choose the "EmpTID" column from the "Column Name:" drop down box. The alias you have assigned will appear in from of the column name.
11. From the "Join Operation" radio button options, choose the equal character (=).

    Your end result for the Join Properties should look like, or close to, the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.



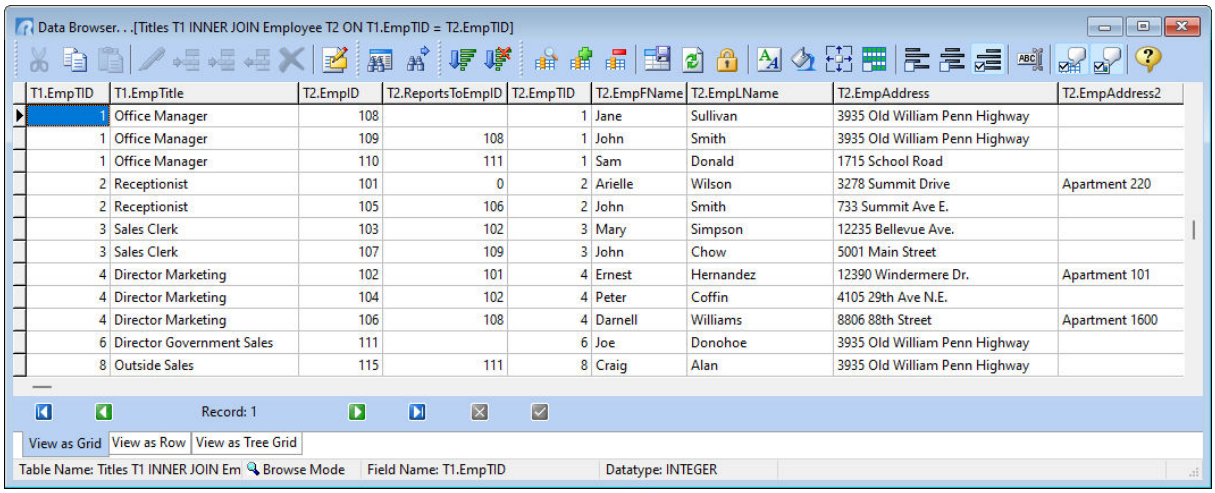
12. Select the "OK" button.

    In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT *
FROM  Titles T1 INNER JOIN Employee t2 ON T1.EmpTID = t2.EmpTID
```

13. Then, browse the query results by selecting "Query" > "Browse Query" from the main Menu Bar. The results should look like the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.



T1.EmpTID	T1.EmpTitle	T2.EmpID	T2.ReportsToEmpID	T2.EmpTID	T2.EmpFName	T2.EmpLName	T2.EmpAddress	T2.EmpAddress2
1	Office Manager	108		1	Jane	Sullivan	3935 Old William Penn Highway	
1	Office Manager	109	108	1	John	Smith	3935 Old William Penn Highway	
1	Office Manager	110	111	1	Sam	Donald	1715 School Road	
2	Receptionist	101	0	2	Arielle	Wilson	3278 Summit Drive	Apartment 220
2	Receptionist	105	106	2	John	Smith	733 Summit Ave E.	
3	Sales Clerk	103	102	3	Mary	Simpson	12235 Bellevue Ave.	
3	Sales Clerk	107	109	3	John	Chow	5001 Main Street	
4	Director Marketing	102	101	4	Ernest	Hernandez	12390 Windermere Dr.	Apartment 101
4	Director Marketing	104	102	4	Peter	Coffin	4105 29th Ave N.E.	
4	Director Marketing	106	108	4	Darnell	Williams	8806 88th Street	Apartment 1600
6	Director Government Sales	111		6	Joe	Donohoe	3935 Old William Penn Highway	
8	Outside Sales	115	111	8	Craig	Alan	3935 Old William Penn Highway	

The results show the "Employee" table added to the "Titles" table. Each employee is displayed next to the title where the "EmpID" column matches.

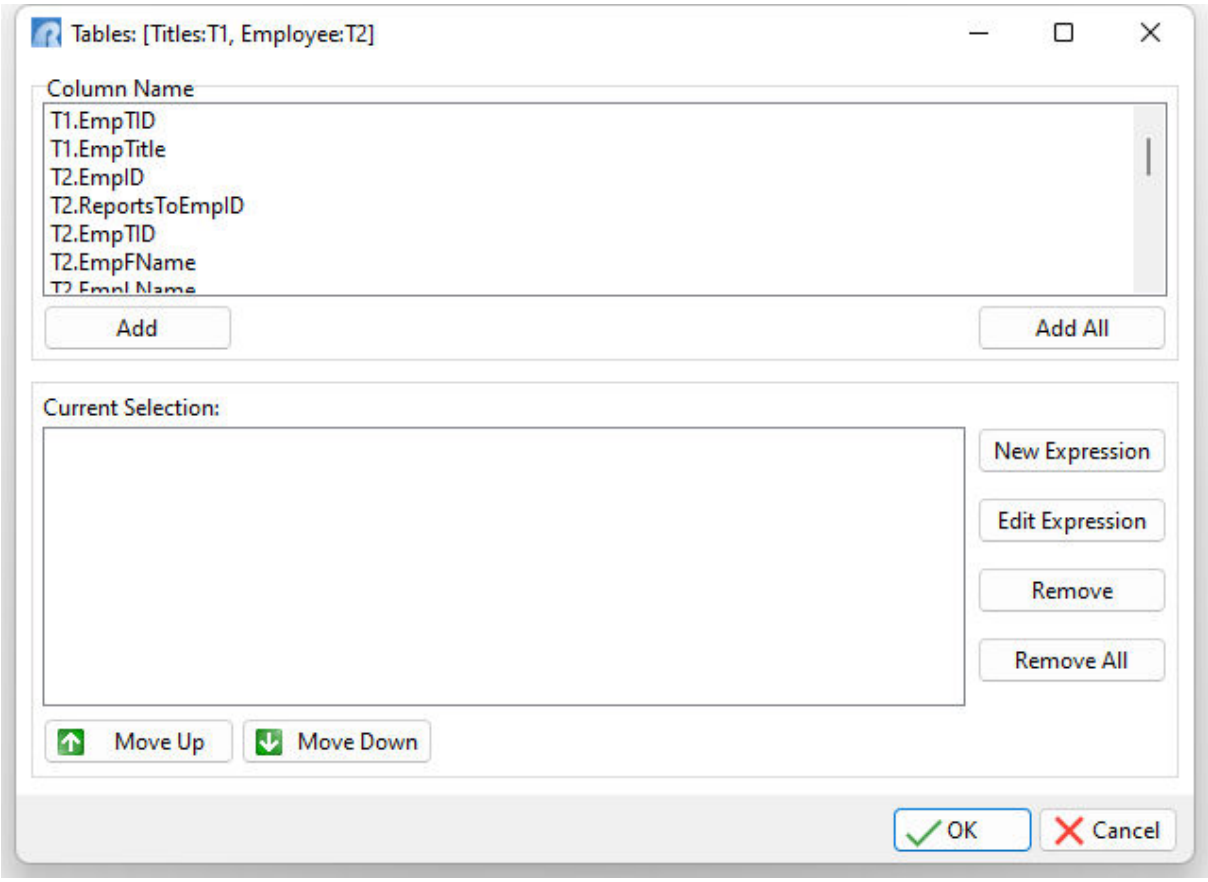
14. Close the Data Browser window.

Next, the number of columns displayed will be decreased.

15. Right click on the "Titles" table under "Tables/Views In Use", and choose "Select Columns..." from the speed menu.

A dialog will be displayed with the available columns for each of the tables.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Once the "Tables" dialog is displayed, you can add the columns one at a time by selecting the desired column and pressing the "Add" button, or double click on a desired column. All columns can be added at once by pressing the "Add All" button.

16. Select the "EmpTitle" column, and select the "Add" button.

When a column is added, an optional column alias can be assigned in the "Column Alias" dialog, that appears each time a column is added to the query/view.

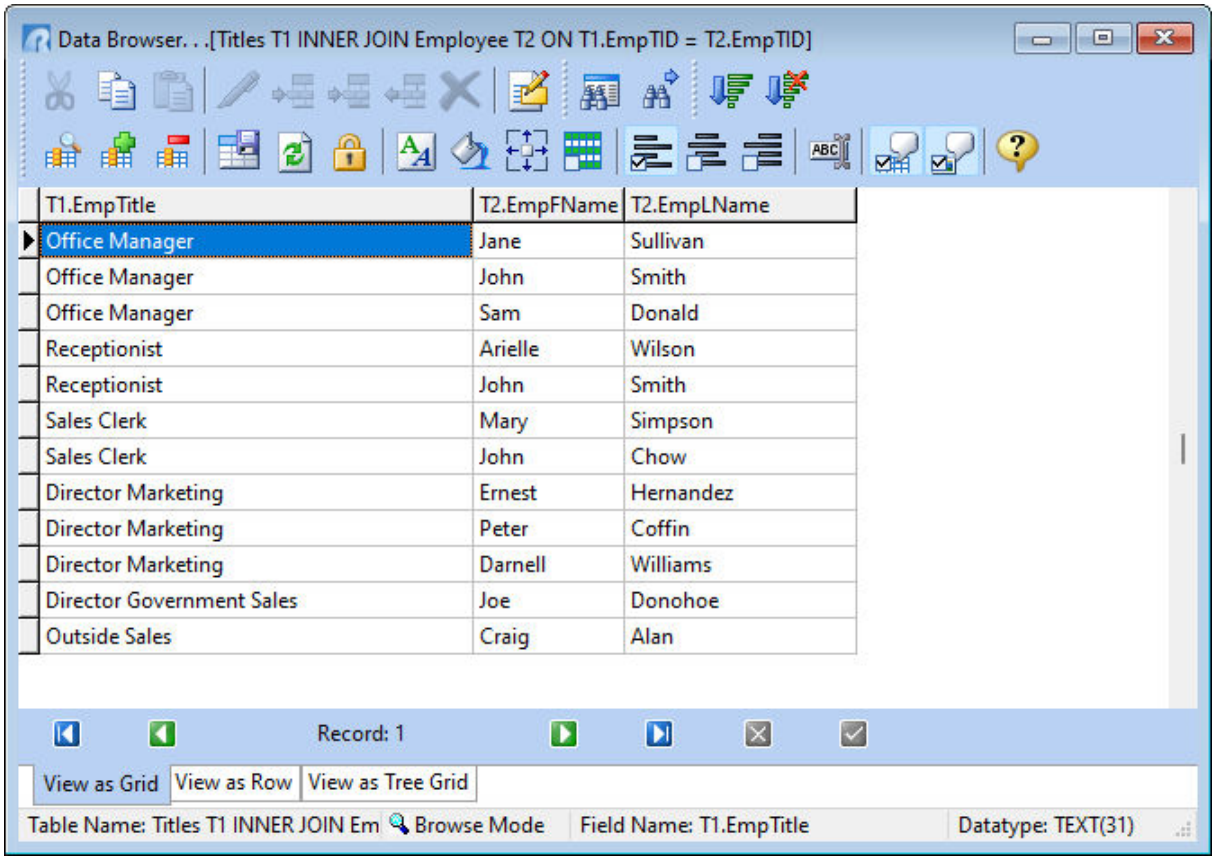
17. Press the "OK" or "Cancel" buttons to allow R:BASE to assign the alias for you.
18. Also add the "EmpFName" and "EmpLName" columns individually and do not enter an alias for each.
19. Press the "OK" button to save the columns added.

In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT T1.EmpTitle,T2.EmpFName,T2.EmpLName
FROM  Titles T1 INNER JOIN Employee T2 ON T1.EmpTID = T2.EmpTID
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

20. Now, browse the query results by selecting "Query" > "Browse Query" from the main Menu Bar. The results should look like the following:



The results show the "Employee" table added to the "Titles" table and only the title first name and last name columns are displayed.Each employee is displayed next to the title where the "EmpID" column matches.

If you wish to save the query, follow the below steps.

1. From the main Menu Bar, select "File" > "Save Query As View..."
2. In the dialog, enter "TitleEmp" within the "View Name:" field.
3. In the dialog, enter "Title and Employee List" within the "View Comment:" field.
4. Select the "OK" button.
5. Close the Query Builder by selecting "File" > "Close" from the Menu Bar.

In the Database Explorer, the view will be displayed with the name, comment, and number of columns.

1.4.2.2 Left Outer Join

When using an Outer Join, rows are not required to have matching values. The table order in the FROM clause specifies the left and right table. For a Left Outer Join, R:BASE uses each value unique to the left (first) table and completes it with nulls for the columns of the right (second) table when the linking columns do not match.

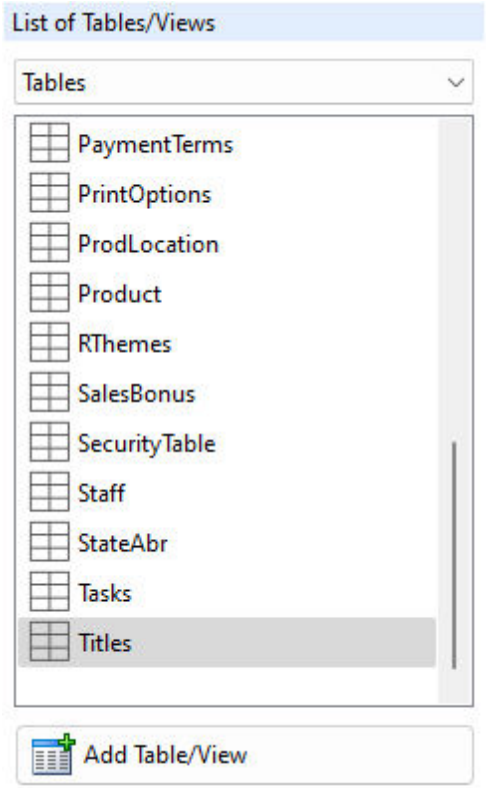
The following instructions will step through the process to recreate a Left Outer Join in the Query Builder.

To continue, launch R:BASE 11 and connect to the RRBYW20 sample database. The database is located in the following default installation directory: "C:\RBTI\RBG11\Samples\RRBYW20".

Copyright © 1982-2024 R:BASE Technologies, Inc.

1. Start R:BASE
2. Connect to the RRBYW20 sample database, by selecting "Database" > "Connect", and navigating to the above default installation directory based upon your version.
3. Then, launch the Query Builder by selecting "Tools" > "Query by Example" from the main Menu Bar

In the Query Builder you will see a list of tables and views within a panel to the left that can be added to your query. The list contains all tables and views for the connected database.



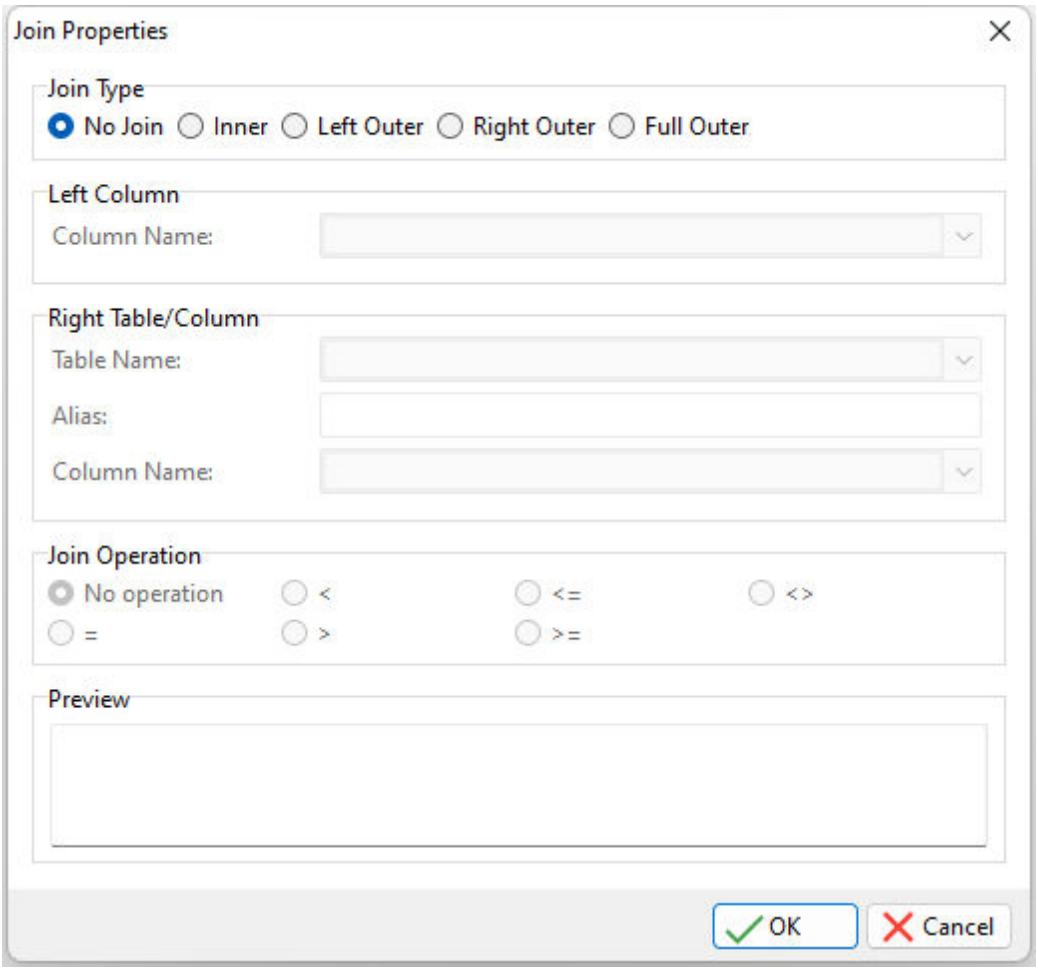
4. Add the "Titles" table to the query by selecting the table, and then selecting the "Add Table/View" button.

The table should now be listed under "Tables/Views In Use".

5. Right click on the "Titles" table under "Tables/Views In Use", and choose "Join Properties..." from the speed menu.

The "Join Properties" dialog will be displayed.

Copyright © 1982-2024 R:BASE Technologies, Inc.



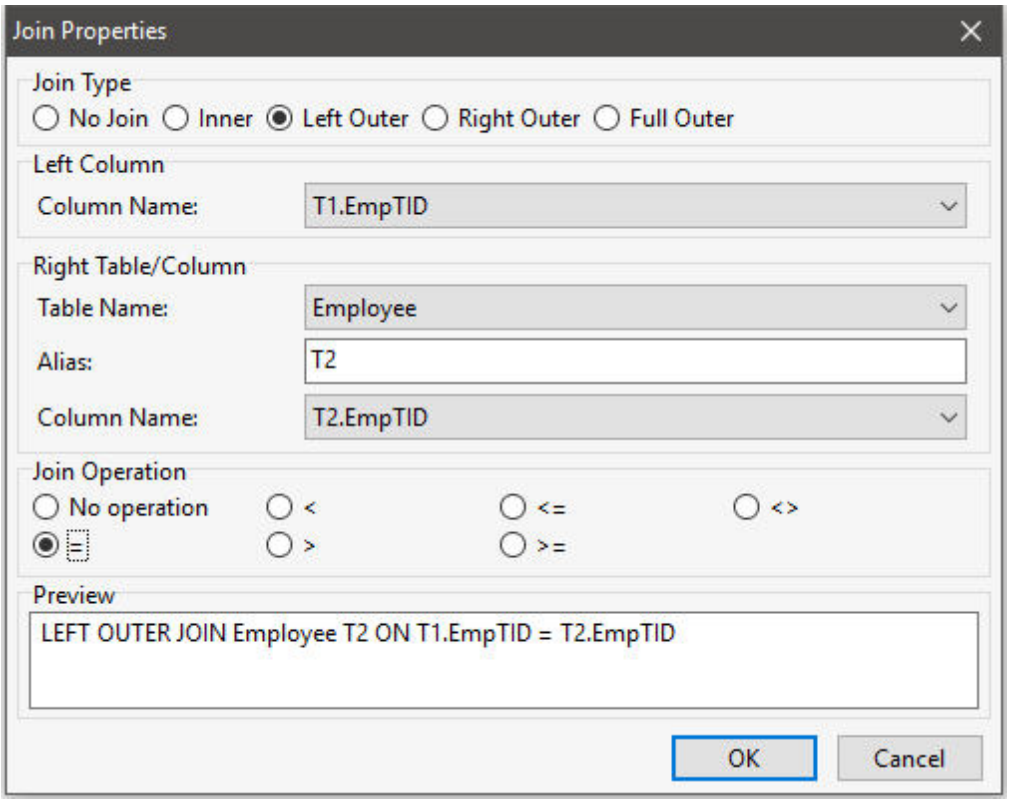
6. From the "Join Type" radio button options, select "Left Outer"

   Take note that the other options within the window will become enabled. In this window you will select the linking columns for the join, the second table for the join, and the join operation. As alias can also be assigned to the linking column.

7. From the "Left Column" panel, choose "T1.EmpTID" from the "Column Name:" drop down box. The "T1" alias may vary from your R:BASE screen.
8. From the "Right Table/Column" panel, choose "Employee" from the "Table Name:" drop down box.
9. From the "Right Table/Column" panel, enter "T2" into the "Alias:" field to assign the table alias for the join. If "T2" is already used for the "Titles" table, then use "T3" as the alias.
10. Again in the "Right Table/Column" panel, choose the "EmpTID" column from the "Column Name:" drop down box. The alias you have assigned will appear in from of the column name.
11. From the "Join Operation" radio button options, choose the equal character (=).

    Your end result for the Join Properties should look like, or close to, the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.



12. Select the "OK" button.

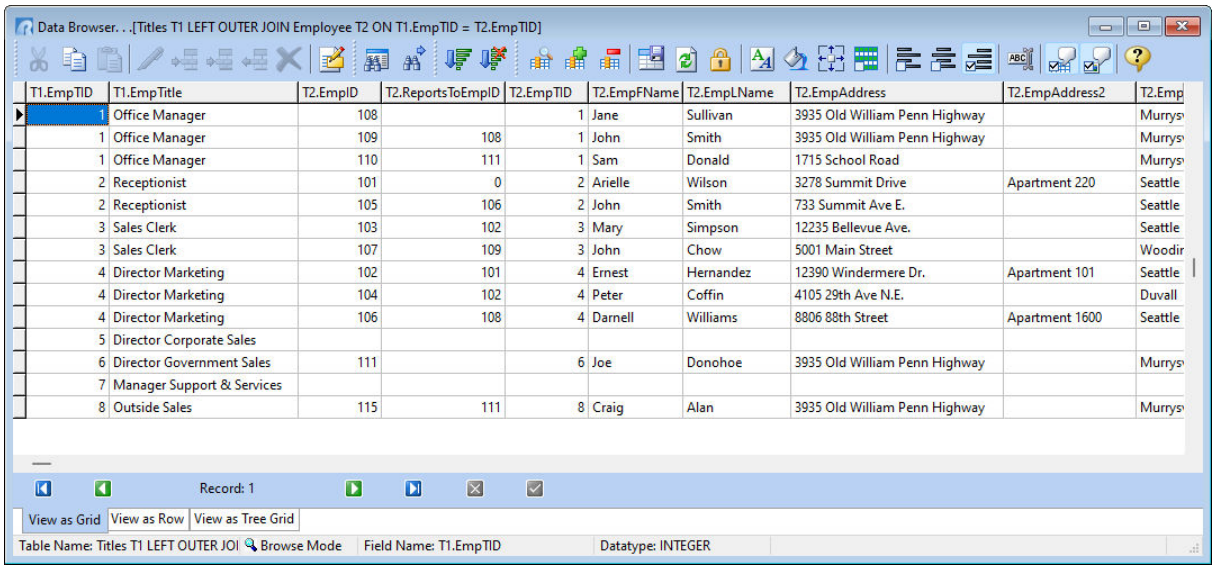
    In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT *
FROM  Titles T1 LEFT OUTER JOIN Employee T2 ON T1.EmpTID = T2.EmpID
```

    The asterisk (*) in the command syntax represents all columns. The next steps will show how to specify certain columns to display.

13. Now, browse the query results by selecting "Query" > "Browse Query" from the main Menu Bar. The results should look like the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.



The results show the Employee table added to the Titles table. Where no employees match for an existing title, the row is empty, or null. In the above, there are no employees that have the "Director Corporate Sales" or "Manager Support & Services" titles.

If you wish to save the query, follow the below steps.

1. From the main Menu Bar, select "File" > "Save Query As View..."
2. In the dialog, enter "TitleEmpList" within the "View Name:" field.
3. In the dialog, enter "Title List with Employees" within the "View Comment:" field.
4. Select the "OK" button.
5. Close the Query Builder by selecting "File" > "Close" from the Menu Bar.

In the Database Explorer, the view will be displayed with the name, comment, and number of columns.

1.4.2.3 Full Outer Join

When using an Outer Join, rows are not required to have matching values. The table order in the FROM clause specifies the left and right table. For a Full Outer Join, R:BASE returns all rows, as long as there's matching data in one of the tables.

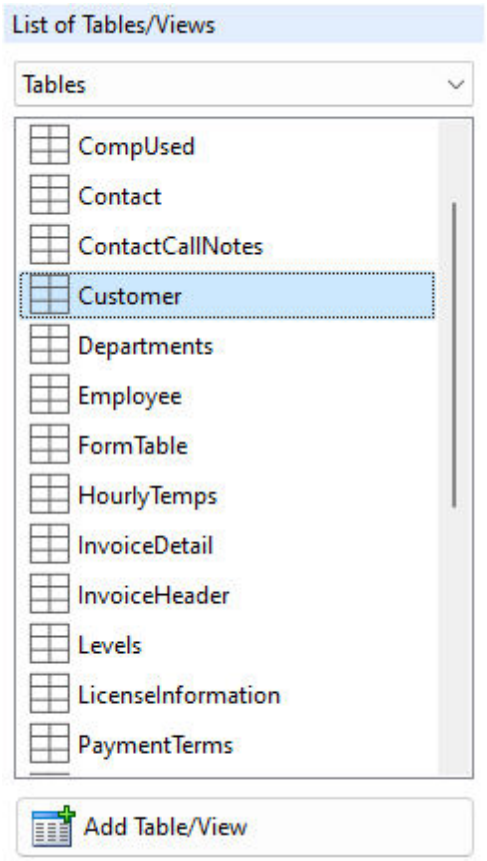
The following instructions will step through the process to recreate a Full Outer Join in the Query Builder.

To continue, launch R:BASE 11 and connect to the RRBYW20 sample database. The database is located in the following default installation directory: "C:\RBTI\RBG11\Samples\RRBYW20".

1. Start R:BASE
2. Connect to the RRBYW20 sample database, by selecting "Database" > "Connect", and navigating to the above default installation directory based upon your version.
3. Then, launch the Query Builder by selecting "Tools" > "Query by Example" from the main Menu Bar

In the Query Builder you will see a list of tables and views within a panel to the left that can be added to your query. The list contains all tables and views for the connected database.

Copyright © 1982-2024 R:BASE Technologies, Inc.



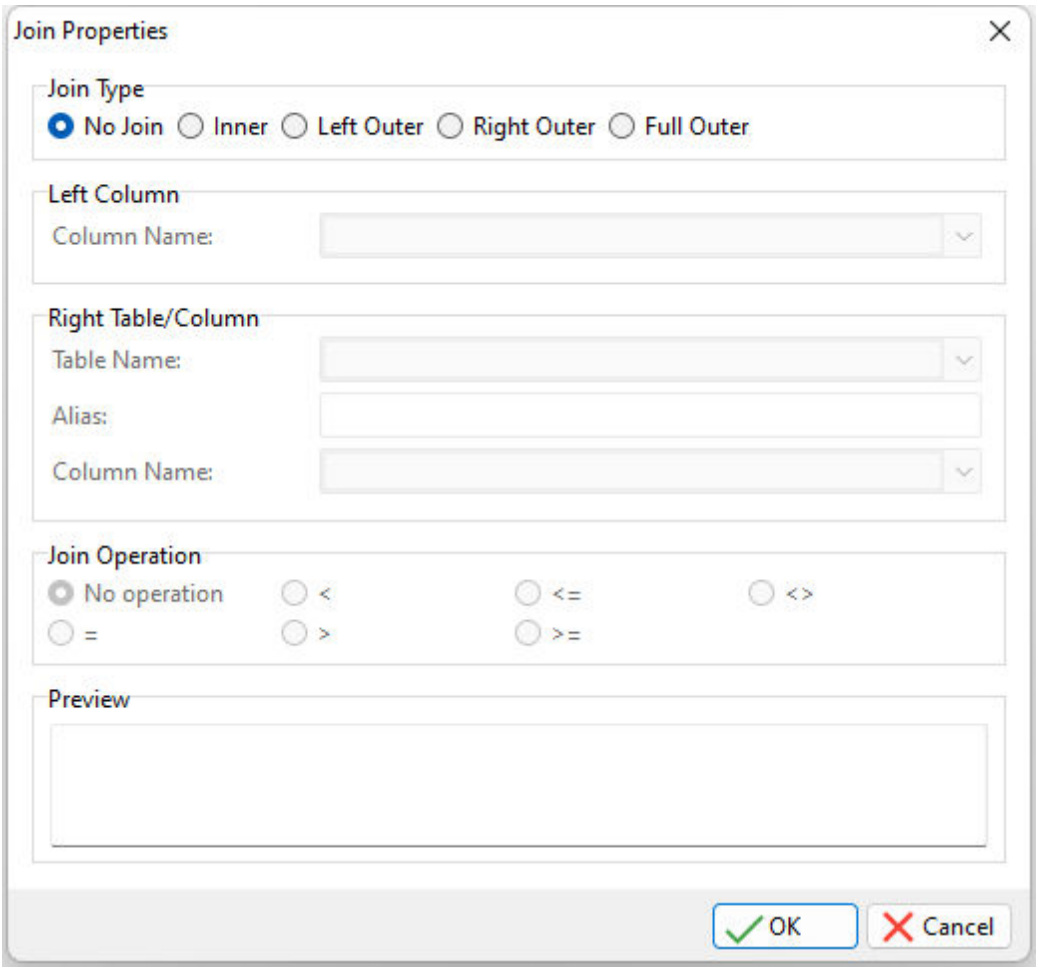
4. Add the "Customer" table to the query by selecting the table, and then selecting the "Add Table/View" button.

   The table should now be listed under "Tables/Views In Use".

5. Right click on the "Customer" table under "Tables/Views In Use", and choose "Join Properties..." from the speed menu.

   The "Join Properties" dialog will be displayed.

Copyright © 1982-2024 R:BASE Technologies, Inc.



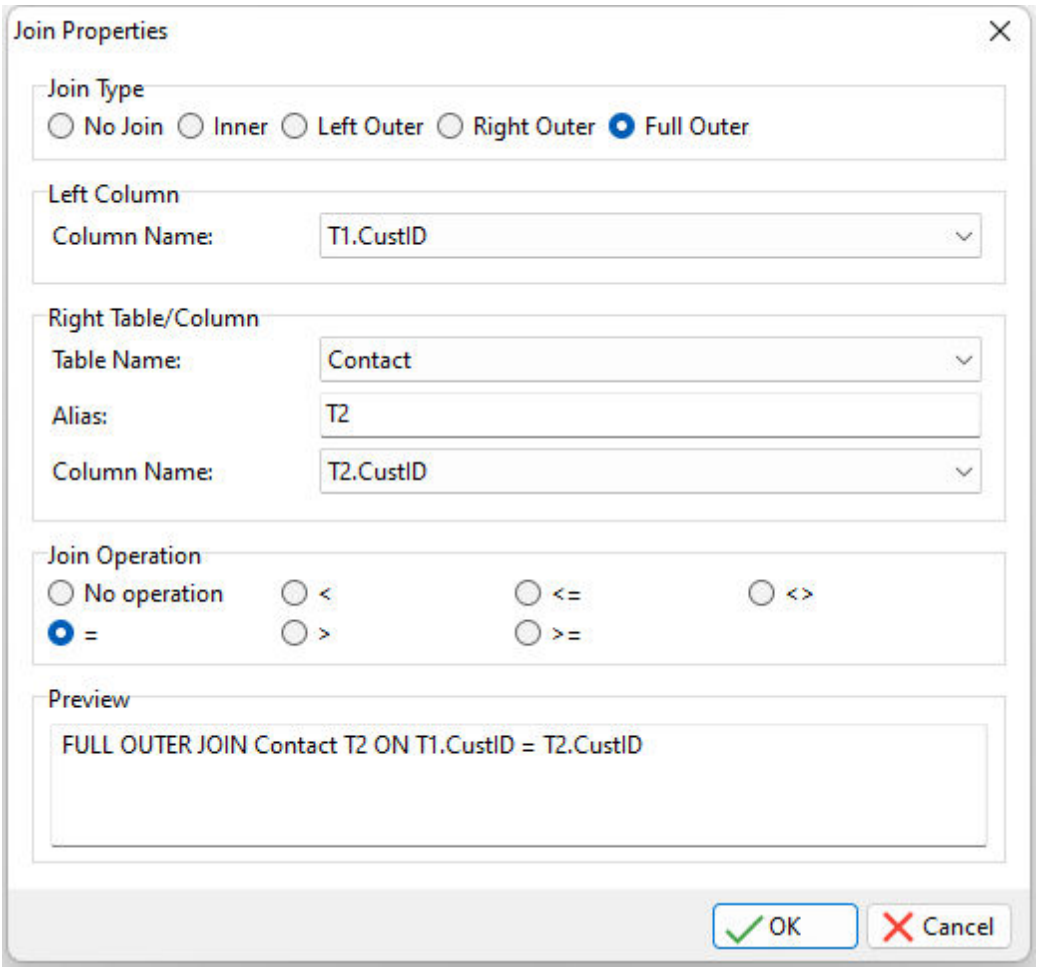
6. From the "Join Type" radio button options, select "Full Outer"

   Take note that the other options within the window will become enabled. In this window you will select the linking columns for the join, the second table for the join, and the join operation. As alias can also be assigned to the linking column.

7. From the "Left Column" panel, choose "T1.CustID" from the "Column Name:" drop down box. The "T1" alias may vary from your R:BASE screen.
8. From the "Right Table/Column" panel, choose "Contact" from the "Table Name:" drop down box.
9. From the "Right Table/Column" panel, enter "T2" into the "Alias:" field to assign the table alias for the join. If "T2" is already used for the "Customer" table, then use "T3" as the alias.
10. Again in the "Right Table/Column" panel, choose the "CustID" column from the "Column Name:" drop down box. The alias you have assigned will appear in from of the column name.
11. From the "Join Operation" radio button options, choose the equal character (=).

    Your end result for the Join Properties should look like, or close to, the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.



12. Select the "OK" button.

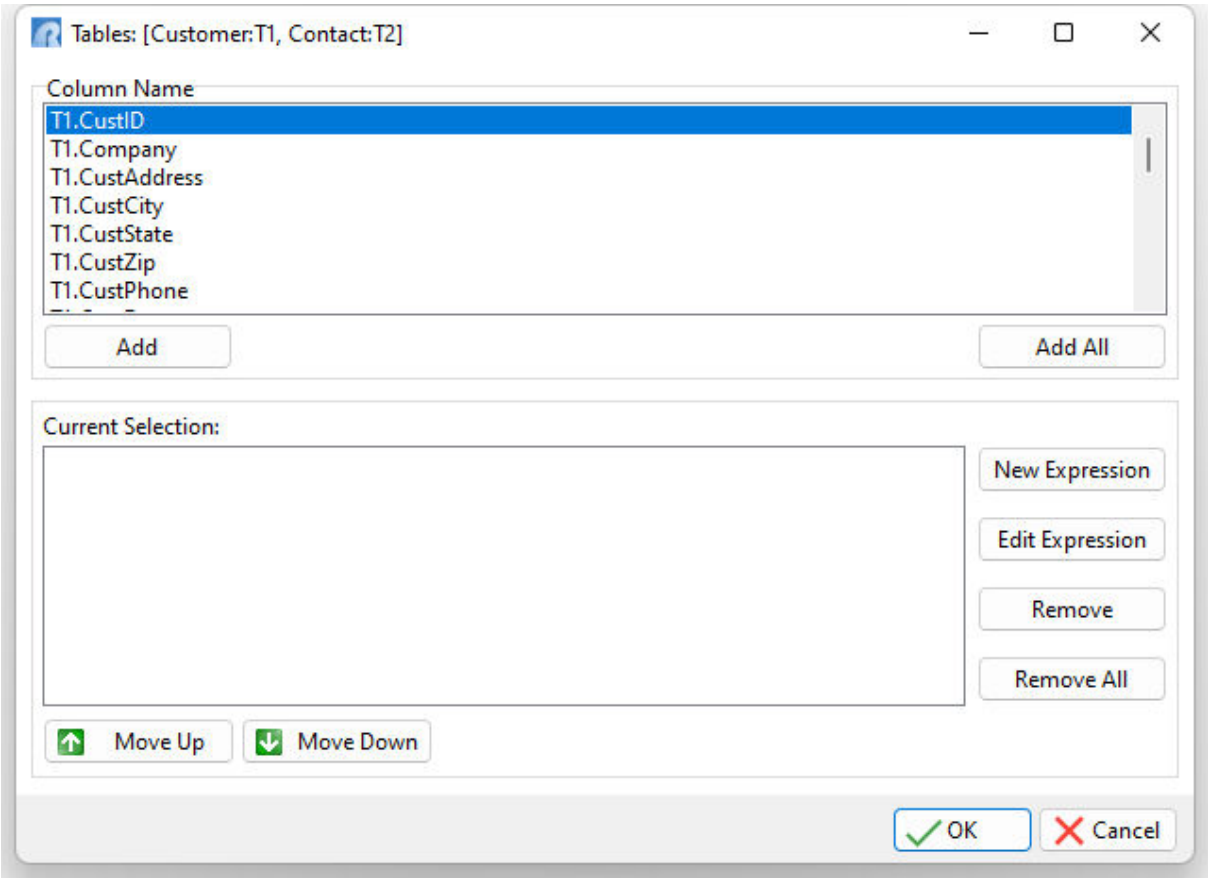
    In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT *
FROM  Customer T1 FULL OUTER JOIN Contact t2 ON T1.CustID = t2.CustID
```

13. Right click on the "Customer" table under "Tables/Views In Use", and choose "Select Columns..." from the speed menu.

    A dialog will be displayed with the available columns for each of the tables.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Once the "Tables" dialog is displayed, you can add the columns one at a time by selecting the desired column and pressing the "Add" button, or double click on a desired column. All columns can be added at once by pressing the "Add All" button.

14. Select the "Company" column, and select the "Add" button.

When a column is added, an optional column alias can be assigned in the "Column Alias" dialog, that appears each time a column is added to the query/view.

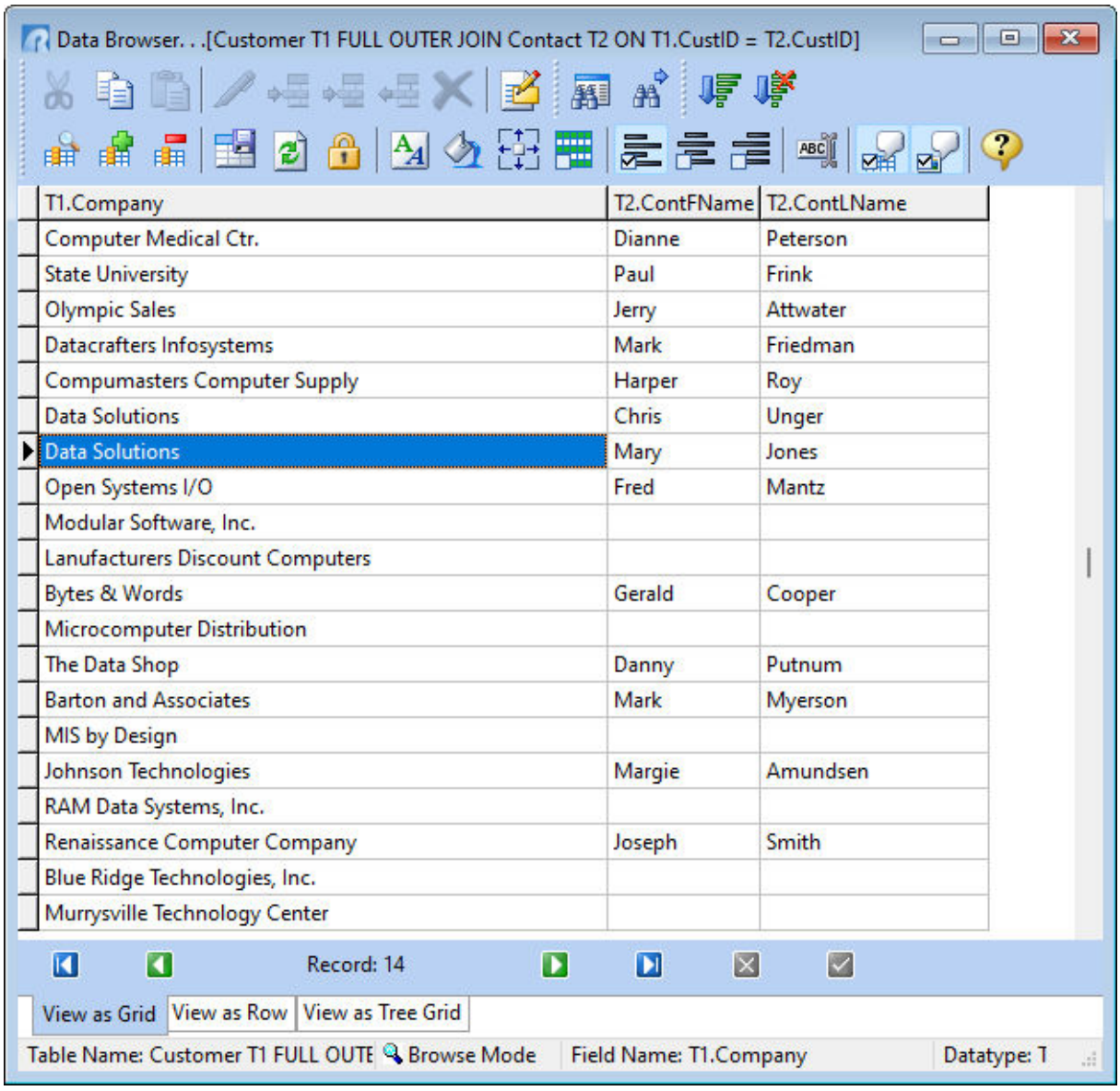
15. Press the "OK" or "Cancel" buttons to allow R:BASE to assign the alias for you.
16. From the "Contacts" table, add the "ContFName" and "ContLName" columns individually, and do not enter an alias for each.
17. Press the "OK" button to save the columns added.

In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT T2.Company,T3.ContFName,T3.ContLName
FROM  Customer T2 FULL OUTER JOIN Contact T3 ON T2.CustID = T3.CustID
```

18. Now, browse the query results by selecting "Query" > "Browse Query" from the main Menu Bar. The results should look like the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.



The results show the Contact table added to the Customer table. Notice that where any Contact matches for an existing Customer, the company is listed. For example, since there are three contacts for the "Computer Warehouse - II" the company name is listed for each contact.

If you wish to save the query, follow the below steps.

1. From the main Menu Bar, select "File" > "Save Query As View..."
2. In the dialog, enter "CustContacts" within the "View Name:" field.
3. In the dialog, enter "Customer and Contact List" within the "View Comment:" field.
4. Select the "OK" button.
5. Close the Query Builder by selecting "File" > "Close" from the Menu Bar.

In the Database Explorer, the view will be displayed with the name, comment, and number of columns.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.4.3 Building Queries Using GROUP BY

A GROUP BY clause groups rows according to the values in one or more columns and sorts the results. GROUP BY consolidates the information from several rows into one row. This results in a table with one row for each value in the named column or columns and one or more values per column. With the GROUP BY the columns can be sorted in ascending or descending order.

The columns listed in the GROUP BY clause are related to those listed in the query's selected columns. Any column named in the GROUP BY clause can also be named in the query's selected columns, but any column not named in the GROUP BY clause can be used only in the selected columns if the column is used in a SELECT function (SUM, COUNT, etc.).

An optional HAVING clause can also be used with a GROUP BY.

HAVING Clause
The HAVING clause determines which rows of data to include based on the results of the GROUP BY clause, and essentially limits the rows affected by the GROUP BY clause.

The HAVING clause selects rows that meet one or more conditions from among the results of the GROUP BY clause. HAVING works the same as a WHERE clause with the following exceptions:

- A WHERE clause modifies the intermediate results of a FROM clause; a HAVING clause modifies the intermediate results of a GROUP BY clause
- A HAVING clause can include SELECT Functions

1.4.3.1 Grouping Employee Job Titles

A GROUP BY can be used to consolidate information from several rows into one row.

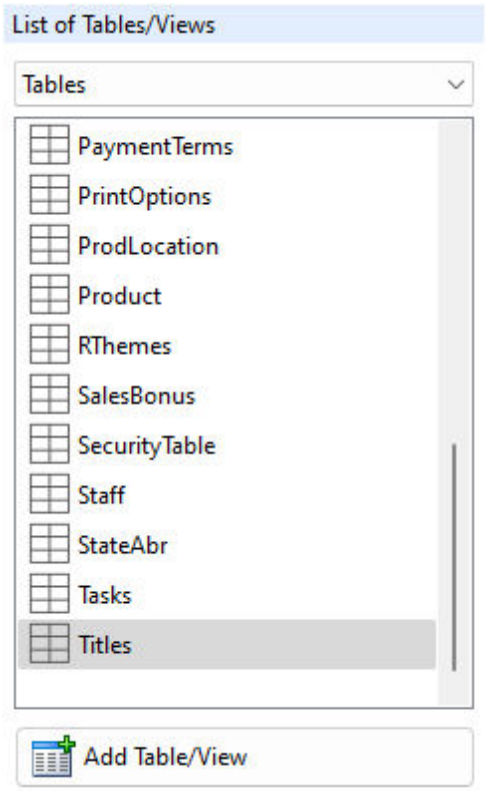
The following instructions will step through the process to recreate a view using a GROUP BY with the COUNT Function in the Query Builder.

To continue, launch R:BASE 11 and connect to the RRBYW20 sample database. The database is located in the following default installation directory: "C:\RBTI\RBG11\Samples\RRBYW20".

1. Start R:BASE
2. Connect to the RRBYW20 sample database, by selecting "Database" > "Connect", and navigating to the above default installation directory based upon your version.
3. Then, launch the Query Builder by selecting "Tools" > "Query by Example" from the main Menu Bar

   In the Query Builder you will see a list of tables and views within a panel to the left that can be added to your query. The list contains all tables and views for the connected database.

Copyright © 1982-2024 R:BASE Technologies, Inc.



4. Add the "Titles" table to the query by selecting the table, and then selecting the "Add Table/View" button.

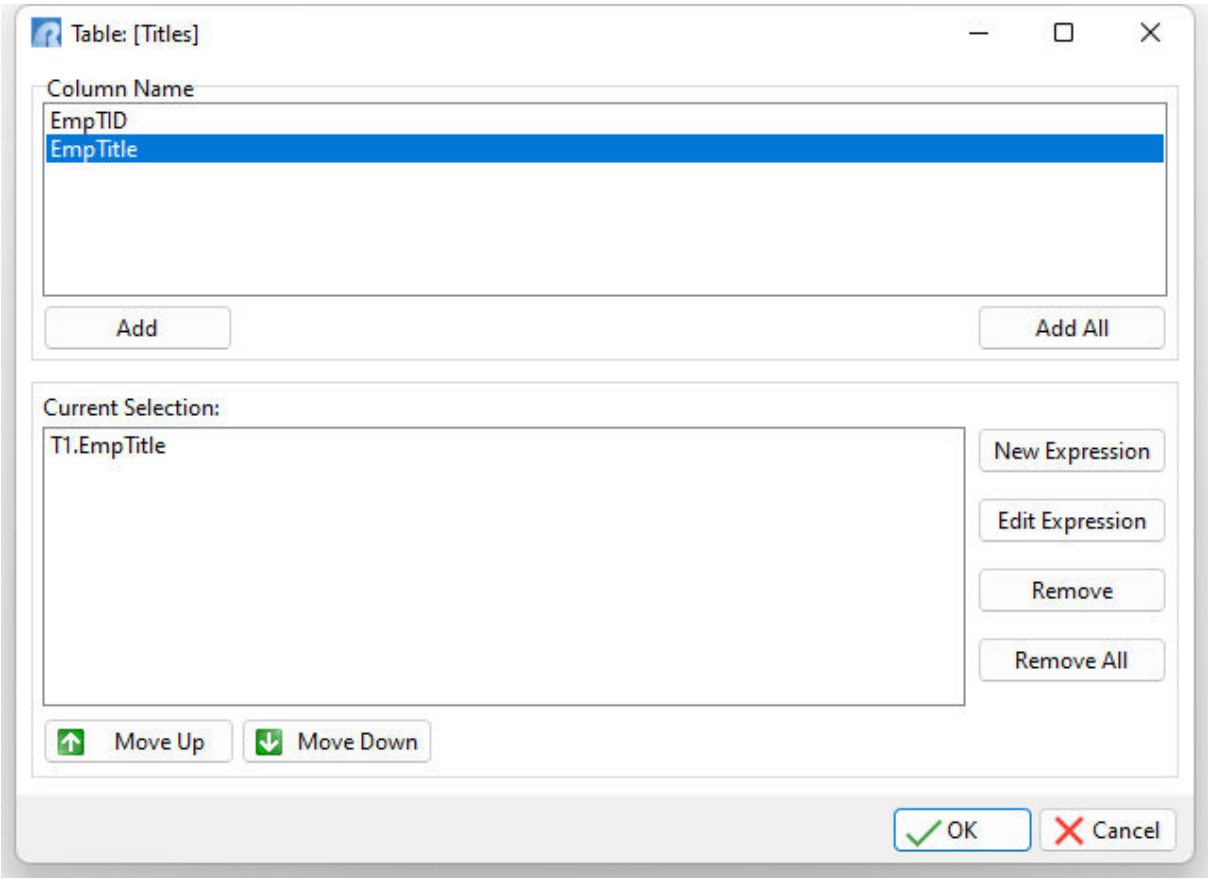
   The table should now be listed under "Tables/Views In Use".

5. Add the "Employee" table to the query by selecting the table, and then selecting the "Add Table/View" button.

   The table should now be listed under "Tables/Views In Use". With the tables added, the specific column(s) to be used in the query can be selected.

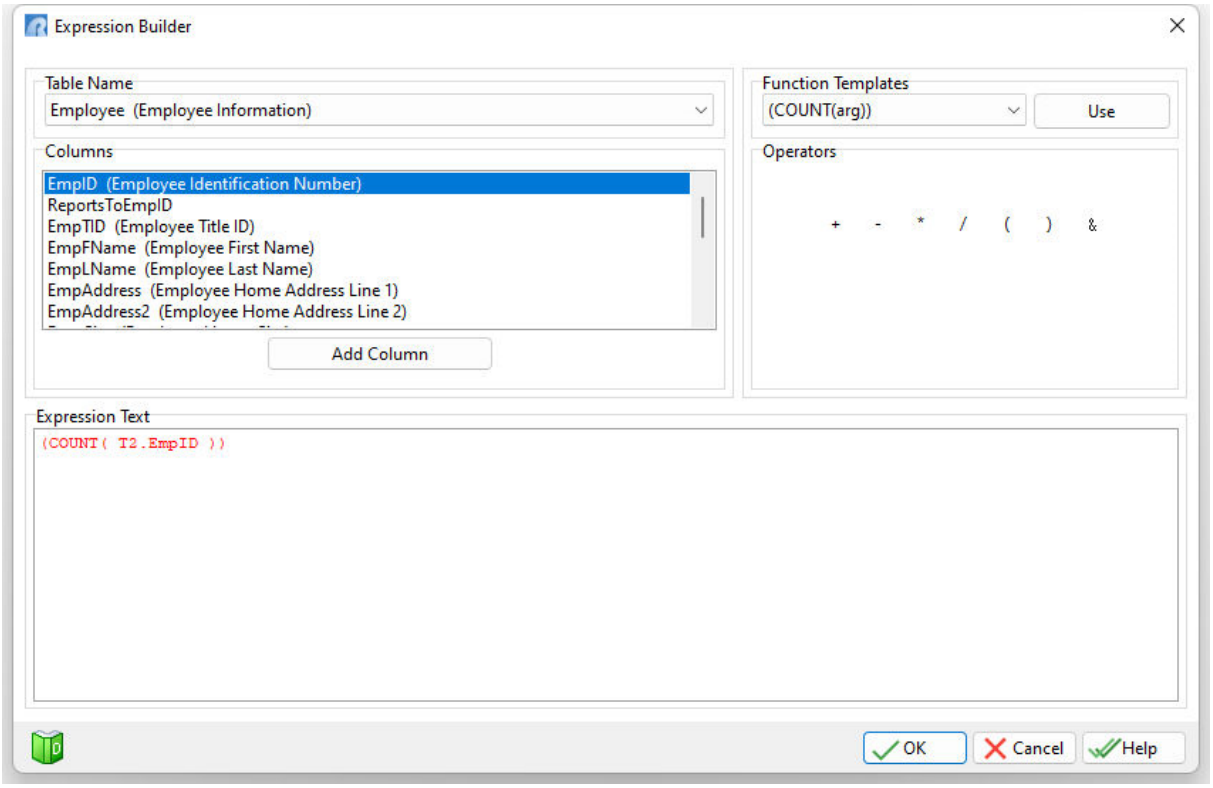
6. Right click on the "Titles" table and select "Select Columns ..."

Copyright © 1982-2024 R:BASE Technologies, Inc.



7. Add the "EmpTitle" column. When prompted for a column alias, just select the OK button.
8. Click on the "New Expression" button.
9. Using the "Table Name" drop down box, select "Employee" as the table name.
10. From the "Function Templates", select "(COUNT(arg))", then the "Use" button. The COUNT function will appear in the "Expression Text" panel.

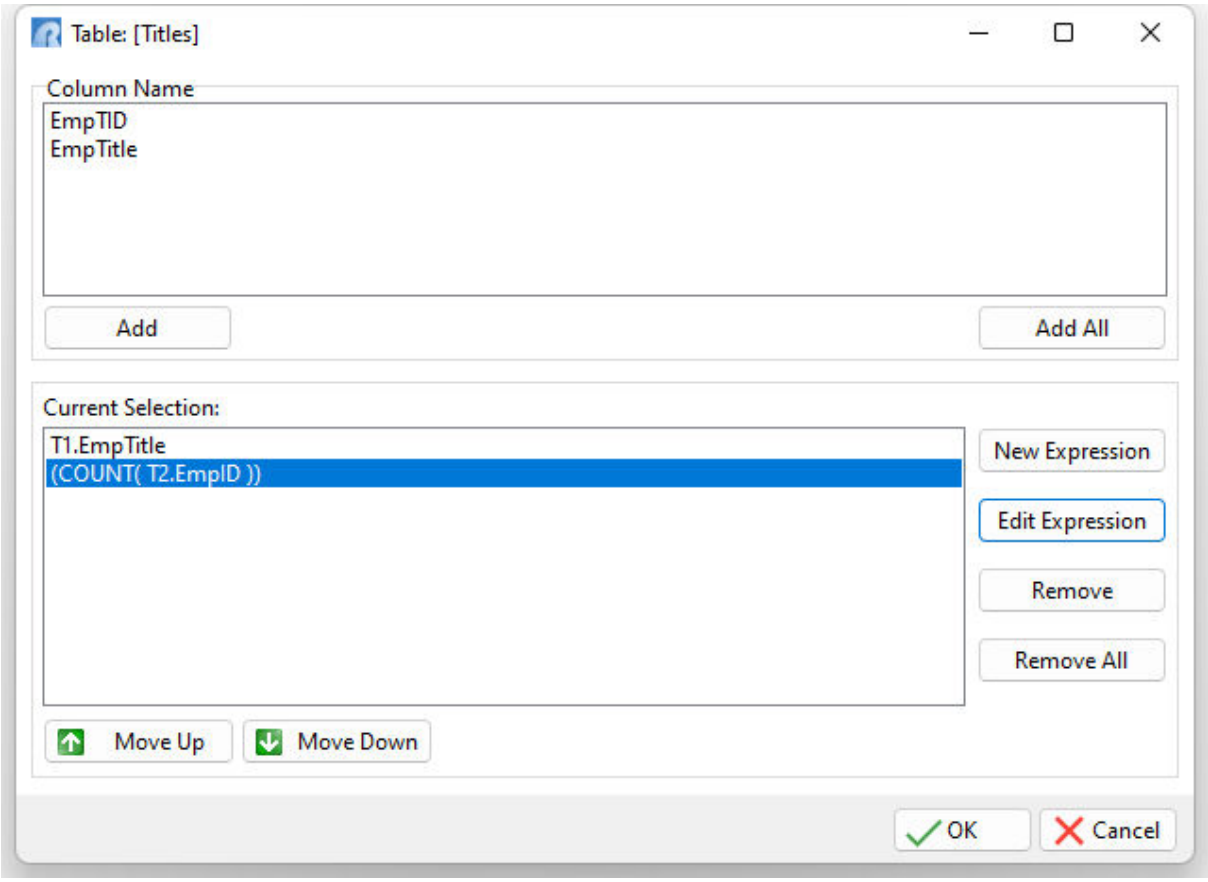
Copyright © 1982-2024 R:BASE Technologies, Inc.



11. Within the "Expression Text" panel, remove the text "arg", and select "EmpID" from the list of "Columns", then select the "Add Column" button.
12. Select the "OK" button.

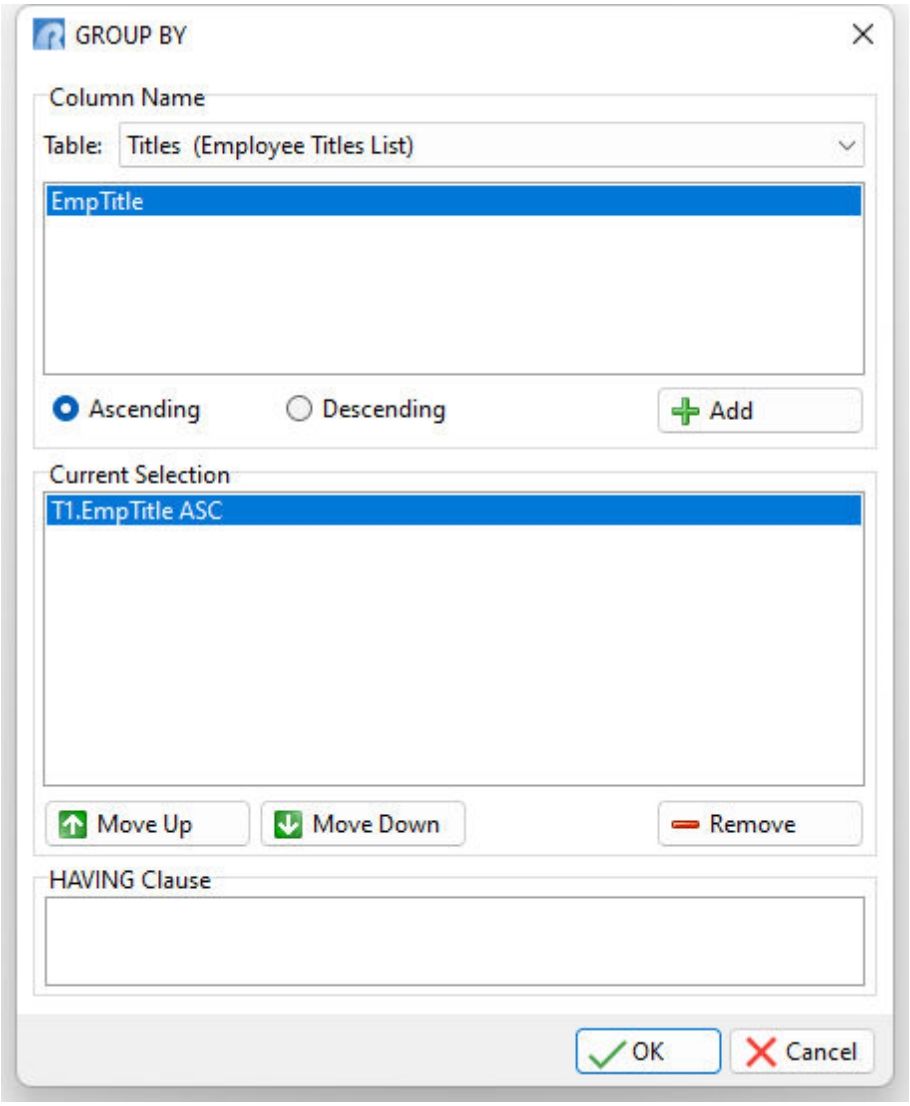
The "Current Selection" results should look like this:

Copyright © 1982-2024 R:BASE Technologies, Inc.



13. Select the "OK" button to return to the Query Builder.
14. Right click on the "Titles" table under "Tables/Views In Use", and choose "GROUP BY" from the speed menu.

Copyright © 1982-2024 R:BASE Technologies, Inc.



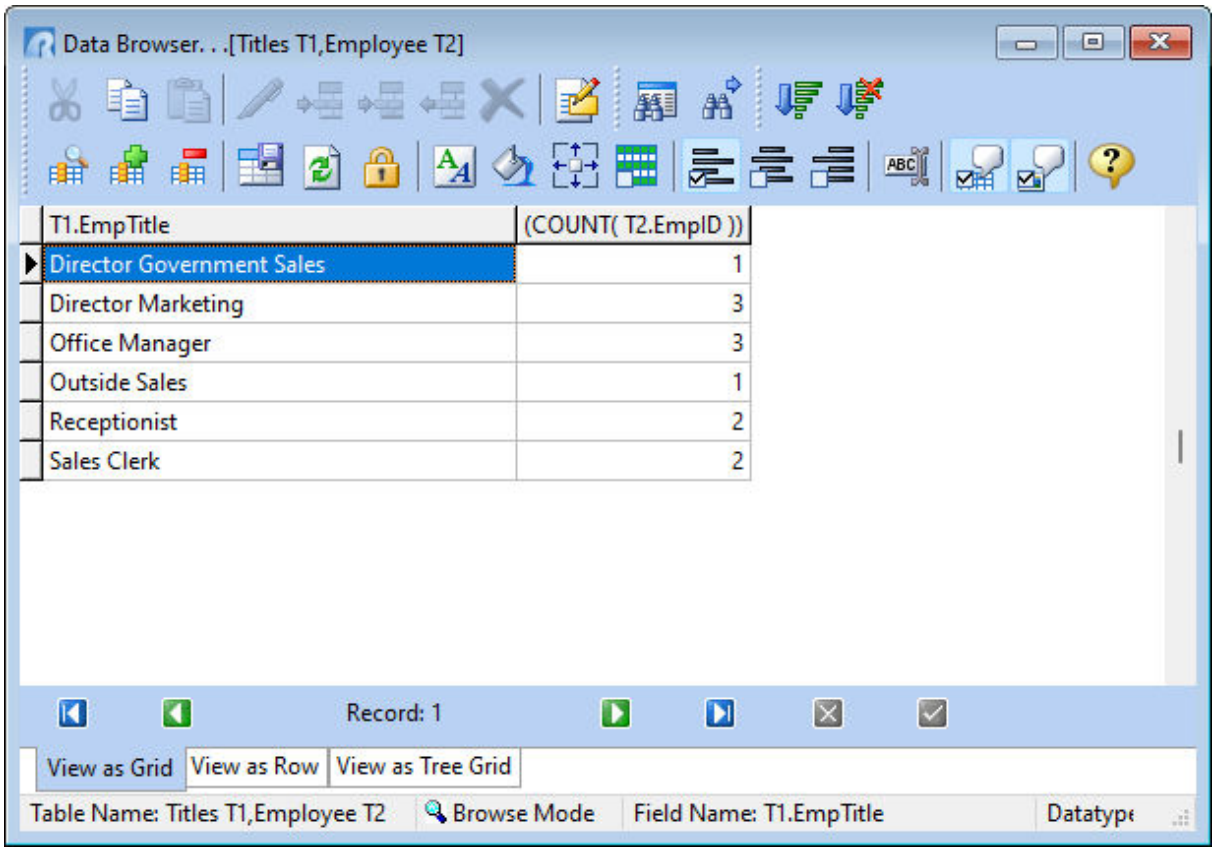
15. Select the "EmpTitle" column, leave the "Ascending" radio button selected, and click the "Add" button.
16. Select the "OK" button to return to the Query Builder.

In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT T1.EmpTitle,(COUNT( T2.EmpID ))
FROM  Titles T1, Employee T2
WHERE T1.EmpTID = T2.EmpTID
GROUP BY T1.EmpTitle
```

17. Now, browse the query results by selecting "Query" > "Browse Query" from the main Menu Bar. The results should look like the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.

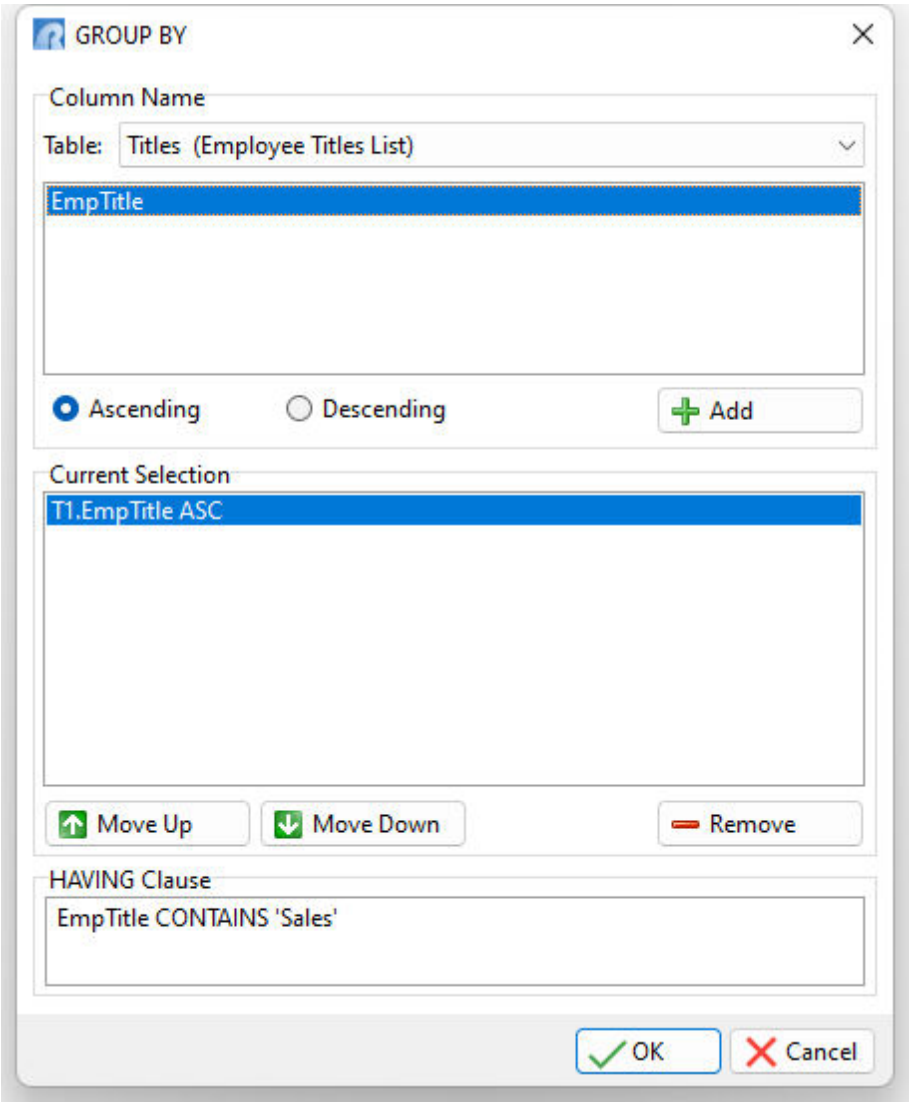


The results should display a column of employee titles, and the number of employees with each title. Close the Data Browser window.

A HAVING clause can be added to only display job titles with "sales" in the name.

18. Right click on the "Titles" table under "Tables/Views In Use", and choose "GROUP BY" from the speed menu.

Copyright © 1982-2024 R:BASE Technologies, Inc.



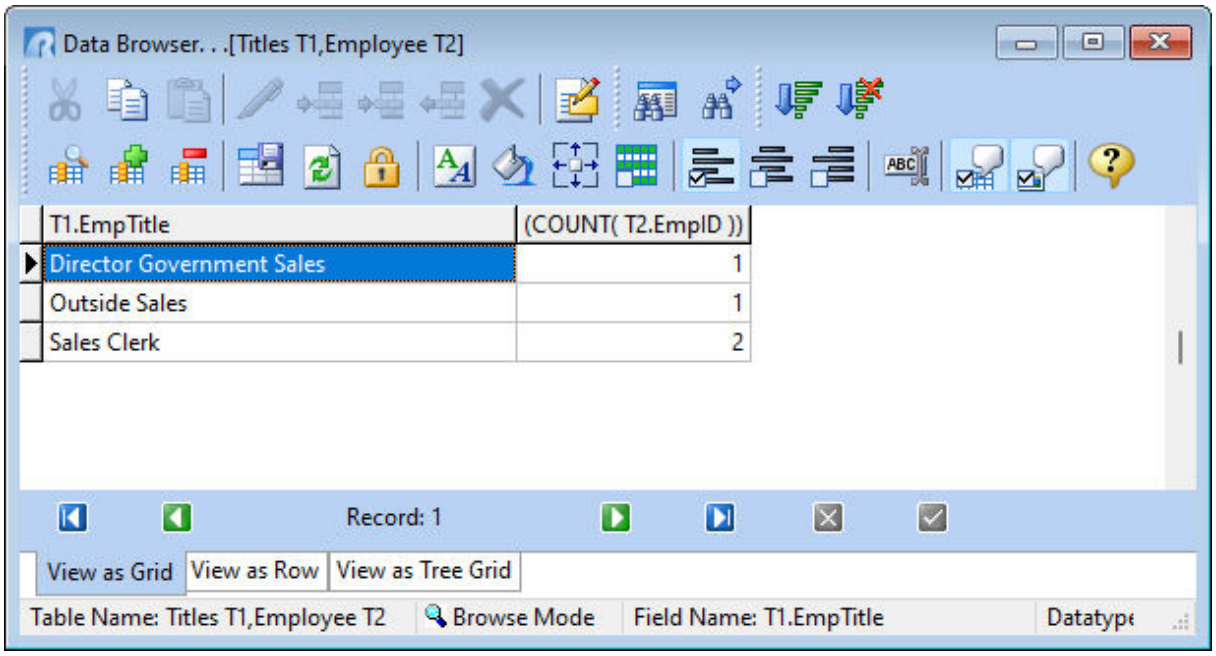
19. Within the "HAVING Clause" panel, enter: EmpTitle CONTAINS 'Sales'
20. Select the "OK" button to return to the Query Builder.

In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT T1.EmpTitle,(COUNT( T2.EmpID ))
FROM  Titles T1, Employee T2
WHERE T1.EmpTID = T2.EmpTID
GROUP BY T1.EmpTitle
HAVING EmpTitle CONTAINS 'Sales'
```

21. Now, browse the query results by selecting "Query" > "Browse Query" from the main Menu Bar. The results should look like the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.

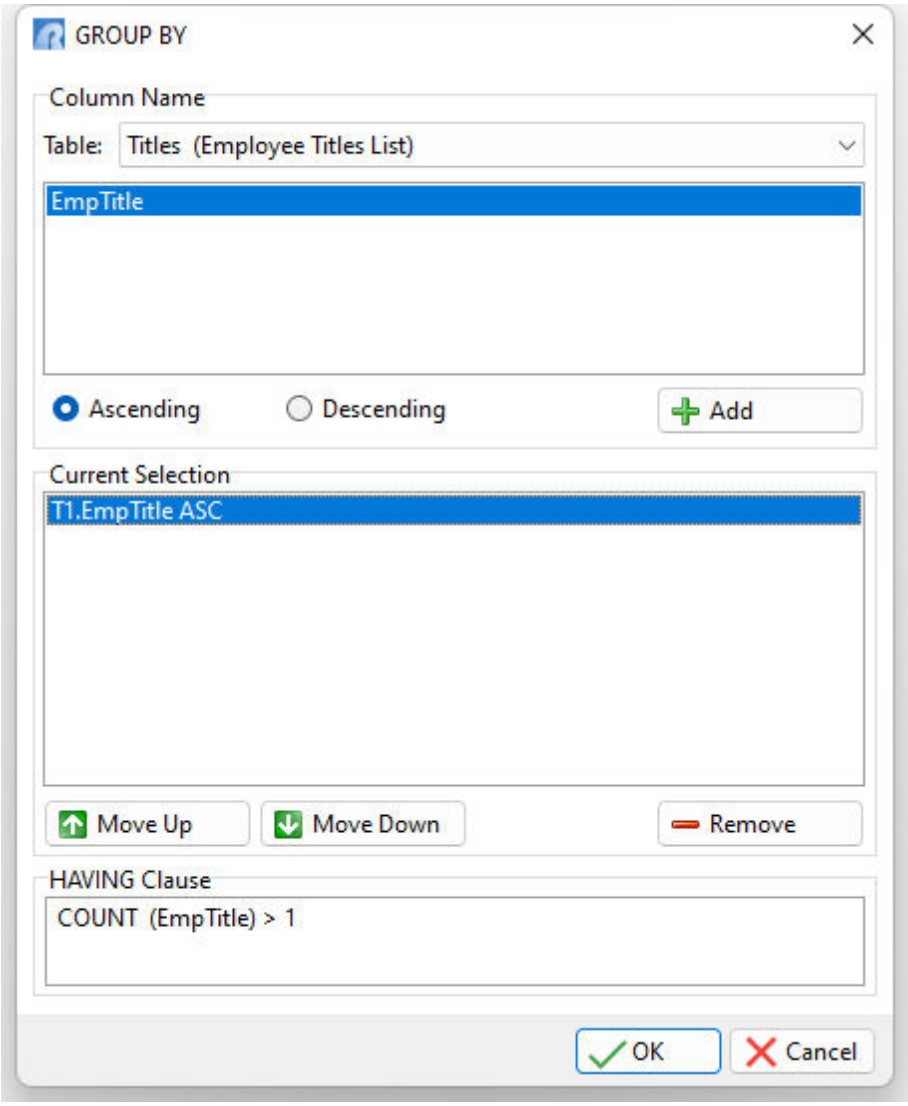


The results should display a column of employee titles that contain the word "sales" in the name, and the number of employees with each title. Close the Data Browser window.

A HAVING clause can also be used to only display limited results based upon employee titles that occur more than once.

22. Right click on the "Titles" table under "Tables/Views In Use", and choose "GROUP BY" from the speed menu.

Copyright © 1982-2024 R:BASE Technologies, Inc.



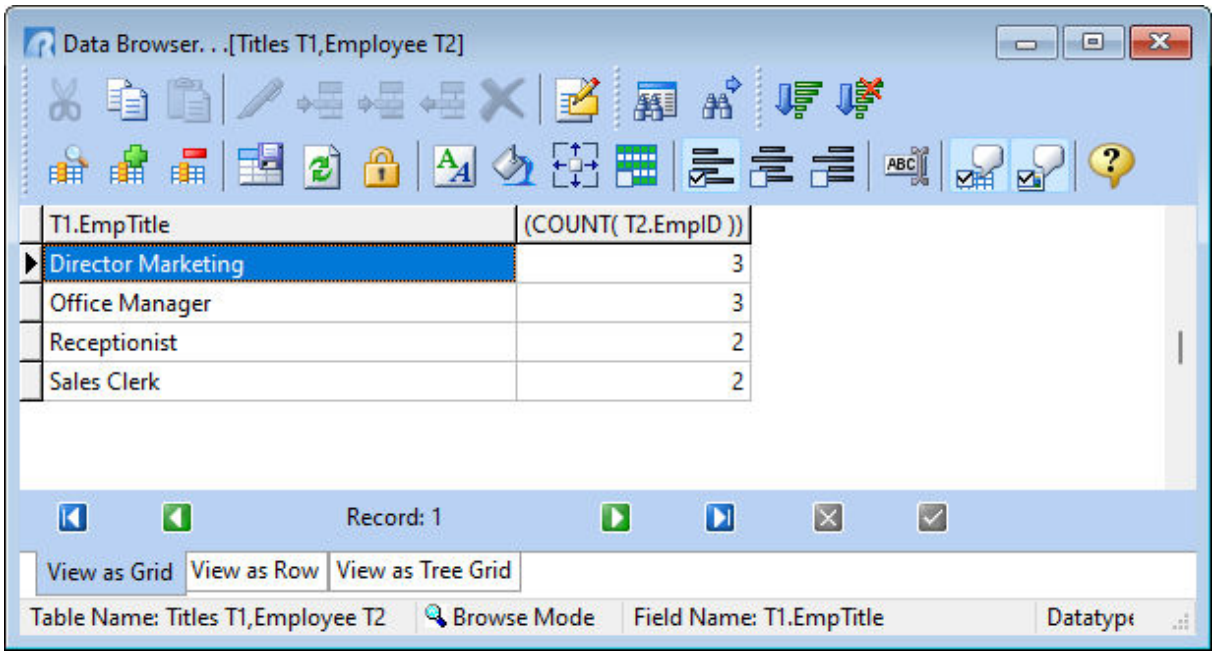
23. Within the "HAVING Clause" panel, enter: COUNT (EmpTitle) > 1
24. Select the "OK" button to return to the Query Builder.

In the Query Builder main window, you should see the SQL syntax displayed in the bottom of the page as:

```
SELECT T1.EmpTitle,(COUNT( T2.EmpID ))
FROM  Titles T1, Employee T2
WHERE T1.EmpTID = T2.EmpTID
GROUP BY T1.EmpTitle
HAVING COUNT (EmpTitle) > 1
```

25. Now, browse the query results by selecting "Query" > "Browse Query" from the main Menu Bar. The results should look like the following:

Copyright © 1982-2024 R:BASE Technologies, Inc.



The results should display a column of employee titles that occur more than once. Close the Data Browser window.

If you wish to save the query, follow the below steps.

1. From the main Menu Bar, select "File" > "Save Query As View..."
2. In the dialog, enter "JobTitle" within the "View Name:" field.
3. In the dialog, enter "Employee Job Titles" within the "View Comment:" field.
4. Select the "OK" button.
5. Close the Query Builder by selecting "File" > "Close" from the Menu Bar.

   In the Database Explorer, the view will be displayed with the name, comment, and number of columns.

1.5 Constraints

To control the data that enters your database, you can apply powerful data-integrity rules called *constraints*. By applying a constraint to a column, you can prevent irreconcilable and empty data from being entered. R:BASE uses the following constraints:

- **Primary Key -** A column or set of columns that uniquely identify a row; in other words, each value in a primary key column is unique. A primary-key constraint prevents duplicate (non-unique) and null values from being entered. Even if you do not specifically define a constraint, all tables (in a well-designed database) should have a primary key. You can define one primary key per table.

- **Foreign Key -** A column or set of columns that match values in a particular primary key or unique key defined in a different table. A value cannot be inserted or changed.

- **Unique Key -** A column or set of columns that uniquely identify a row; in other words, each value in a unique-key column is unique. A unique-key constraint prevents duplicate (non-unique) and null values from being entered. The only difference between a unique key and a primary key is that you can define multiple unique keys per table.

- **Unique Index** - A column or set of columns that uniquely identify a row, but cannot be referenced like a primary key or unique key. The differences between a unique key and a unique index is that the unique key must be defined a Not NULL.

Copyright © 1982-2024 R:BASE Technologies, Inc.

- **Not NULL** - Placing a not null constraint on a column requires that the data in the column must contain a value, and cannot be null. This prevents users from adding a "blank" value. A not null constraint cannot be added if the column already contains null values.

Constraints cannot be turned off and are always enforced; you must delete the constraint if you do not want it. However, because R:BASE works with constraints faster, use constraints instead of rules when possible.

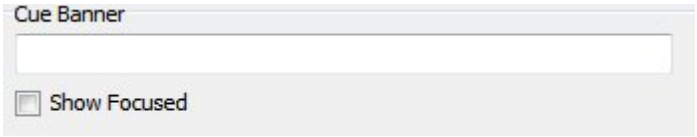
You can remove constraints from columns. If you want to remove a primary- or unique-key constraint, you must first remove the foreign-key constraints that refer to it.

See also:

ALTER TABLE
CREATE INDEX
SET FASTFK
Data Designer

1.6 Cue Banner Display

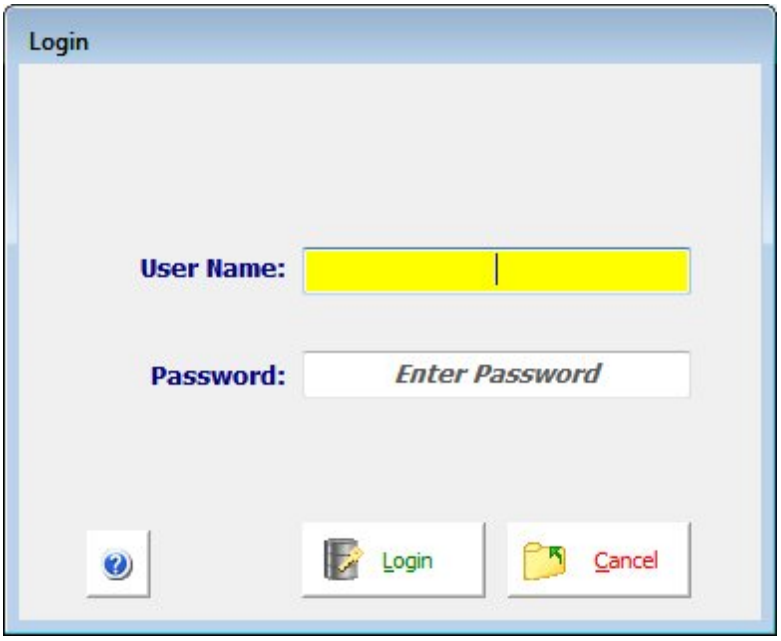
The cue banner displays a faint, or grayed, text message that appears in a field. The cue banner is displayed within the field until the cursor focus lands on the field or when keystrokes are entered, after which the message disappears. The cue banner can be used with DB Edit and Variable Edit form controls, and the settings are available on the "Additional" tab of the object properties. A message and show focused setting are available for the cue banner. An example for is provided within the RRBYW20 sample database. Locate and run the "CueBanner" form.

⇨ **Cue Banner** Specifies a text hint to appear in the field, which disappears when the focus lands on the field *Show Focused* Specifies if the hint remains until keystrokes are pressed (Windows Vista, 7, and higher only)	

In the following sample form, the cue banner "Enter User Name" and "Enter Password" are displayed in the edit fields.

Copyright © 1982-2024 R:BASE Technologies, Inc.

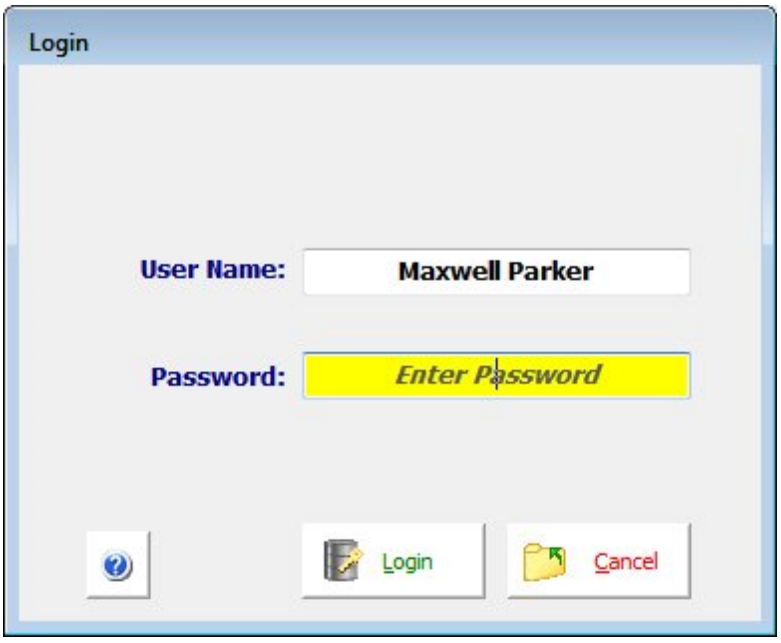
Once the focus lands on the "User Name" field, the cue banner disappears.



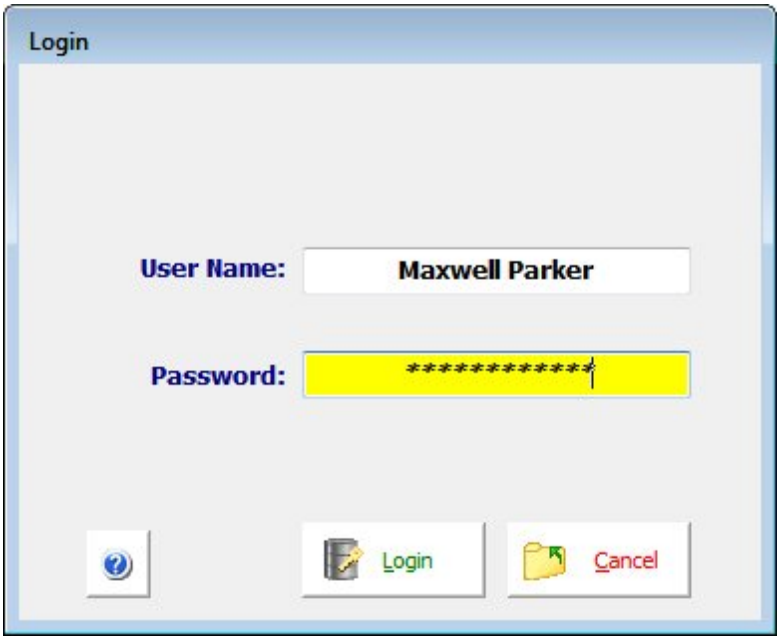
After entering text and moving to the next field, the cue banner does not return for the "User Name" field.

In the "Password" field, the cue banner is set to remain until keystrokes are pressed for the field. This optional setting in only support with Windows Vista, 7, and higher.

Copyright © 1982-2024 R:BASE Technologies, Inc.



After entering text, the cue banner disappears.



The PROPERTY command can also be used to specify the cue banner.

```
PROPERTY DBUserName CUEBANNER 'Enter User Name'
PROPERTY DB_Password CUEBANNER 'Enter Password'
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.7 Cursors Explained

A cursor is a valuable programming tool. It is a pointer to rows in a table. A cursor lets you step through rows one by one, performing the same action on each row. You can set a cursor to point to all the rows in a table or to a subset of rows. A cursor is set using the DECLARE CURSOR command.

The DECLARE CURSOR command does not work by itself, but is really a sequence of commands. In addition to the DECLARE CURSOR, the OPEN and FETCH commands are required. A WHILE loop is used to step through the rows and perform the programmed action on each row. The CLOSE or DROP command is used after the cursor has stepped through all the rows.

The basic sequence of commands for a cursor is as follows:

```
DECLARE c1 CURSOR FOR +
   SELECT custid, company FROM customer
OPEN c1
FETCH c1 INTO vcustid1 INDI ind1, vcompany INDI ind2
WHILE SQLCODE <> 100 THEN
   -- Place code for row by row actions here.
   FETCH c1 INTO vcustid1 INDI ind1, vcompany INDI ind2
ENDWHILE
DROP CURSOR c1
```

The DECLARE CURSOR command names the cursor and defines the set of rows. The cursor name is then used in the OPEN, FETCH, CLOSE, and DROP commands that reference it. A cursor name can be up to 18 characters long and follows the same naming conventions as all other names in R:BASE.

More than one cursor can be defined and open at a time. SELECT is used in the DECLARE CURSOR to identify the rows to step through. The SELECT part of a cursor declaration can point to rows from a single table or from multiple tables, and can choose all or only some of the columns from a table. You can use the GROUP BY clause as well as the WHERE and ORDER BY clauses of SELECT.

The OPEN command initializes the cursor and tells R:BASE you are ready to retrieve a row of data from the cursor. The OPEN command positions the cursor at the first row of the set of data defined by the SELECT in the cursor declaration.

The FETCH command retrieves a row of data into the specified variables. The number of variables must match the number of columns listed in the SELECT part of the DECLARE CURSOR command. Each variable has a corresponding indicator variable, which tells if a NULL value was retrieved. The list of variable pairs - data variable and indicator variable - is separated by commas.

The FETCH command sets SQLCODE, the SQL error variable. If a row was retrieved, SQLCODE is set to 0. After the last row is retrieved, FETCH sets SQLCODE to 100 - no more data. Using SQLCODE as the condition for the WHILE loop lets you easily retrieve and act on each successive row. Placing a second FETCH command immediately before the ENDWHILE command keeps fetching rows until the end of data is reached. Then the loop exits.

Within the WHILE loop, place whatever commands are needed to operate on each row. You can look up additional data, perform mathematical calculations, update data, and so on.

When the cursor completes and the WHILE loop is exited, the cursor is dropped with the DROP CURSOR command. A cursor name must be dropped before it can be declared again. DROP removes a cursor definition from memory; to use the cursor again, it must be declared with the DECLARE CURSOR command. CLOSE leaves a cursor definition in memory; to use the cursor again, it is opened with the OPEN command. After a cursor has been closed, an OPEN repositions the pointer at the first row of the cursor definition. CLOSE is most often used with nested cursors, DROP with individual cursors.

When a cursor is open, you can use a special WHERE clause option, WHERE CURRENT OF cursorname. This WHERE clause works with the UPDATE, DELETE, and SELECT commands to perform the specified action on the row the cursor is currently pointing at. The DELETE deletes the entire row; the SELECT, and UPDATE only operate on columns included in the SELECT part of the DECLARE CURSOR command. Note that not every cursor definition supports use of the WHERE CURRENT OF cursorname.

Copyright © 1982-2024 R:BASE Technologies, Inc.

It is not required to use the WHERE CURRENT OF cursorname in your WHERE clause. A WHERE clause that explicitly points to a row of data using values stored in variables can be used. The unique row identifier is fetched into a variable, then that value is used to access rows in the cursor table or other tables.

```
-- The special WHERE clause WHERE CURRENT OF
-- points to the current row of the cursor.
SELECT custid, company FROM customer +
WHERE CURRENT OF c1

UPDATE customer SET custid = (custid + 1000) +
   WHERE CURRENT OF c1

DELETE FROM customer WHERE CURRENT OF c1

-- Alternatively, use an explicit WHERE
-- clause to access a row.

SELECT custid, company FROM customer +
   WHERE custid = .vcustid

UPDATE customer SET custid = (custid + 1000) +
   WHERE custid = .vcustid

DELETE FROM customer WHERE custid = .vcustid
```

This is the basic cursor structure. Other types of cursors and cursor structures that are used are: multi-table cursors, non-updatable cursors, nested cursors, resettable cursors, and scrolling cursors. Each is briefly described below.

1.7.1 Multi-Table Cursors

A multi-table cursor includes more than one table in the SELECT part of the cursor declaration. The tables can be linked directly within the DECLARE CURSOR command; avoiding steps to define a view to retrieve data from more than one table.

The DECLARE CURSOR command has the full capabilities of the SELECT command to do multi-table queries. As with the SELECT command itself, you list the columns to retrieve, the tables to get the data from, then link the tables in the WHERE clause. For example,

```
-- Select data from both the Customer
-- and Transmaster tables.
DECLARE C1 CURSOR FOR SELECT +
custid, company, transid, transdate, invoicetotal +
   FROM customer, transmaster +
   WHERE customer.custid = transmaster.custid
OPEN C1

-- The fetch retrieves all the specified columns into variables.
FETCH C1 INTO vcustid1 INDI ind1, vcompany INDI ind2 +
   vtransid INDI ind3, vtransdate INDI ind4, vinvoicetotal INDI ind5
WHILE SQLCODE <> 100 THEN

   -- Place code for row by row actions here.
   -- An explicit WHERE clause must be used,
   -- WHERE CURRENT OF is not supported with
   -- multi-table cursors.
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
   -- Get the next row
   FETCH C1 INTO vcustid1 INDI ind1, vcompany INDI ind2 +
      vtransid INDI ind3, vtransdate INDI ind4, vinvoicetotal INDI ind5
ENDWHILE
DROP CURSOR C1
```

Notice that the basic structure of the cursor commands doesn't change. You still declare the cursor, open it, fetch the first row, then use a WHILE loop to step through each row. There is no limit to the number of tables that can be included in a DECLARE CURSOR command. The tables are joined together in the same way they are joined with a regular SELECT command.

A multi-table cursor definition is a non-updatable cursor, however. You cannot update the cursor directly by using WHERE CURRENT OF cursorname. You must use explicit WHERE clauses to access the cursor tables.

1.7.2 Non-Updatable Cursors

A non-updatable cursor is one that does not support use of the special WHERE clause WHERE CURRENT OF cursorname. An explicit WHERE clause must be used to access data in the tables.

A non-updatable cursor is a multi-table cursor, or a cursor that is defined, for example, using the GROUP BY clause. The SELECT command that defines the cursor rows does not allow the cursor to point back to a single specific row in a table.

Non-updatable cursors are a very useful part of the DECLARE CURSOR structure. Use the power of the SELECT command in the DECLARE CURSOR declaration to dramatically improve the performance of a cursor. The more work the cursor does, the less your program has to do and the faster and more efficiently it will run.

When using a non-updatable cursor, make sure you fetch a unique row identifier for use in WHERE clauses.

1.7.3 Nested Cursors

A nested cursor involves two DECLARE CURSOR definitions. The second cursor is dependent on the first and its cursor definition uses a variable value fetched by the first cursor.

There is a specific structure recommended for nested cursors - a row is retrieved from cursor one, then the matching rows in cursor two are retrieved and stepped through. Then the next row is retrieved from cursor one and its matching rows from cursor two are stepped through. The process continues until all rows have been retrieved from cursor one.

Example

```
-- The DECLARE commands are done together
-- at the top of the program.
-- An OPEN cursor does not need to immediately
-- follow the corresponding DECLARE CURSOR command
SET VAR vcustid INTEGER
DECLARE c1 CURSOR FOR SELECT custid, company +
   FROM customer ORDER BY company
-- The second cursor uses a variable in the
-- WHERE clause. This variable, vcustid, must be
-- defined earlier in the program.
-- The cursor retrieves rows for a single customer only
DECLARE c2 CURSOR FOR +
   SELECT custid, contfname, contlname +
   FROM contact WHERE custid = .vcustid
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
-- Cursor c1 is opened and the first row retrieved
-- from the Customer table
OPEN c1
FETCH c1 INTO vcustid1 INDI ind1, vcompany INDI ind2
WHILE SQLCODE <> 100 THEN

   -- Cursor c2 is opened, it points to all the
   -- rows in the Contact table that match the
   -- custid fetched into vcustid by cursor c1.
   OPEN c2

   -- Get the first row from the contact table and step
   -- through all matching rows.
   FETCH c2 INTO vcustid1 INDI ind1, vfirstname INDI ind2, +
      vlastname INDI ind3
   WHILE SQLCODE <> 100 THEN

      -- Place code here to do row by row actions

      --Get the next row for cursor c2
      FETCH c2 INTO vcustid1 INDI ind1, vfirstname INDI ind2, +
         vlastname INDI ind3
   ENDWHILE

   -- After all the matching rows in the contact table
   -- have been processed, close cursor c2 and get the
   -- next row from the Customer table.
   -- Cursor c2 is closed and not dropped because
   -- the definition will be reused for the next
   -- row from cursor c1.
   CLOSE c2

   -- Get the next row for cursor c1
   FETCH c1 INTO vcustid1 INDI ind1, vcompany INDI ind2
ENDWHILE

-- Both cursors are dropped when all the rows
-- in the Customer table have been retrieved.
DROP CURSOR c2
DROP CURSOR c1
```

You can use the same WHILE loop condition, SQLCODE <> 100, for both cursors. This works very well and there is no conflict between the two loops. The relative FETCH command sets the value of SQLCODE. Notice that the FETCH from cursor c2 is right before the ENDWHILE of the inner WHILE loop ensuring that that FETCH command is the one being tested by the WHILE loop. The FETCH from cursor c1 is right before the ENDWHILE of the outer WHILE loop, which then continues based on cursor c1. This placement of the DECLARE, OPEN, FETCH, WHILE, and ENDWHILE statements will always work. Just make sure the ENDWHILE is the next command after the FETCH.

With nested cursors, the inner cursor is closed and opened so that it always references the matching rows from the outer cursor. An alternative to opening and closing the inner cursor is to use the RESET option on the OPEN command.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.7.4 Resettable Cursors

A DECLARE CURSOR can use a variable in its WHERE clause. Each time the cursor is opened, the WHERE clause is reevaluated using the current variable value and identifies a new set of data.

You can CLOSE and OPEN a defined cursor, or use the OPEN cursorname RESET command. Don't use the CLOSE command if you place the RESET option on the OPEN command. The RESET option automatically reevaluates the variable value and identifies a new set of data for the cursor.

OPEN cursorname RESET is commonly used with nested cursors. The second cursor is dependent on a variable fetched by the first cursor. By using RESET, you won't need to CLOSE the inner cursor each time.

Using the RESET option on OPEN is faster using than the OPEN, CLOSE sequence of commands.

1.7.5 Scrolling Cursors

Normally, cursors move through the data in one direction only, from top to bottom. They move forward one-by-one through the set of defined rows. Once a row has been accessed and passed over, you can't get back to it. The rows can be ordered in the cursor definition - the top to bottom order is not necessarily the table order.

When a cursor is defined as a scrolling cursor, you gain the capability of moving both forwards and backwards through the rows of data and can also jump past rows.

To define a cursor as a scrolling cursor, include the word SCROLL in the DECLARE CURSOR command. For example,

```
DECLARE c1 SCROLL CURSOR FOR SELECT ...
```

The word SCROLL comes right after the cursor name. If SCROLL is not included in the cursor definition, the cursor can only move forward through the rows one at a time.

Once a cursor is defined as a scrolling cursor, a number of additional options on the FETCH command become available. These options are as follows; note that the directions and positions are based on the order of the rows as specified by the DECLARE CURSOR command, not on the order of the rows in the actual table:

NEXT - The default option if none is specified on the FETCH command. NEXT moves the cursor forward through the rows, it gets the next available row based on the current cursor position. NEXT steps through the rows one-by-one going forward.

PRIOR - Moves the cursor backwards through the rows. The PRIOR option gets the previous row based on the current cursor position, and steps through the rows one-by-one going backwards.

FIRST - Moves the cursor from its current position to the first row. This option jumps immediately to the first row as determined by the DECLARE CURSOR command. A FETCH NEXT then finds the second row. The cursor is repositioned at the beginning of the set of rows.

LAST - Moves the cursor from its current position immediately to the last row as specified by the DECLARE CURSOR command. A FETCH PRIOR then finds the next to last row; a FETCH NEXT returns "end of data encountered". LAST jumps over the rows between the current cursor position and the last row.

ABSOLUTE n - Moves the cursor the specified number of rows from the first row of data as determined by the DECLARE CURSOR and OPEN commands. A positive number must be specified; you can't use this option to move backwards. The intervening rows are jumped over. You can't jump past the last row; if the number given is greater than the number of rows retrieved, an "end of data" error is returned.

RELATIVE n - Moves the cursor the specified number of rows from the current cursor position. This option moves the cursor either forwards or backwards - forwards if a positive number is specified, backwards if a negative number is specified. The intervening rows are jumped over. You can't jump past the last row or the first row; an "end of data" error is returned if the specified number would take you past the beginning or end of the selected rows.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Example
To see how a scrolling cursor can be used in an application, imagine you have a group of customers to contact each day. The scrolling cursor retrieves the list of customers for today. They are ordered by company name. The first row is brought up in a menuless form. The form remains on the screen when you are done with the record, and a CHOOSE menu pops up giving the user choices as to which record to select next.

You can: move through the list of customers one-by-one, both forwards and backwards, jump to the last record and back to the first record, jump past a group of records, and search for a particular record by last name or by company name

Each time you select a record, the cursor is repositioned ready for the next selection.

```
--WALKLIST.RMD
--scroll through a list of customers
SET MESSAGE OFF
SET ERROR MESSAGE OFF
SET VAR vCheckCursor INTEGER = (CHKCUR('c1'))
IF vCheckCursor = 1 THEN
  DROP CURSOR c1
ENDIF
CLS

--Define the scrolling cursor
DECLARE C1 scroll CURSOR FOR +
   SELECT CustId, LastName, Company FROM Customer +
   WHERE calldate = .#DATE ORDER BY Company

--Open the cursor and get the first row
OPEN C1
FETCH FIRST FROM C1 INTO +
   VCustId INDI ICustId, VLastname INDI ILastname, VCompany INDI
   ICompany
WHILE SQLCODE <> 100 THEN

   --Bring up the form with the data from the first row.
   --After the form is closed, choose from the menu which record to
   retrieve next
   EDIT USING CustForm WHERE CustId = .VCustId
   CHOOSE VAction FROM #LIST +
      'Next Customer,Previous Customer,Jump Forward "n",Jump Backward
      "n",+
      Last Customer,First Customer,Search by Last Name,Search by
      Company' +
      TITLE 'Select Customer' CAPTION 'Choose' LINES 8 FORMATTED
   IF VAction = '[Esc]' THEN
      RETURN
   ENDIF

   --The switch/case block determines which record to retrieve
   SWITCH (.VAction)

   --Move forward one row at a time
   CASE 'Next Customer'
      FETCH NEXT FROM C1 INTO +
         VCustId INDI ICustId, VLastname INDI ILastname, +
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
      VCompany INDI ICompany

   --If already on the last row, stay there
   IF SQLCODE = 100 THEN
      FETCH LAST FROM C1 INTO +
      VCustId INDI ICustId, VLastname INDI ILastname, +
      VCompany INDI ICompany
   ENDIF
   BREAK

--Move backward one row at a time
CASE 'Previous Customer'
   FETCH PRIOR FROM C1 INTO +
      VCustId INDI ICustId, VLastname INDI ILastname, +
      VCompany INDI ICompany

   --If already on the first row, stay there
   IF SQLCODE = 100 THEN
      FETCH FIRST FROM C1 INTO +
      VCustId INDI ICustId, VLastname INDI ILastname, +
      VCompany INDI ICompany
   ENDIF
   BREAK

--Move forward the specified number of records
CASE 'Jump Forward "n"'
   DIALOG 'How many to jump forward?' VNum=4 VEndKey 1
   SET VAR VPlus = (INT(.VNum))

   --R:BASE counts from the current cursor position
   FETCH RELATIVE .vplus FROM C1 INTO +
      VCustId INDI ICustId, VLastname INDI ILastname, +
      VCompany INDI ICompany

   --If the number of records to jump past takes you beyond the last
   --record, the last record is retrieved
   IF SQLCODE = 100 THEN
      FETCH LAST FROM C1 INTO +
         VCustId INDI ICustId, VLastname INDI ILastname, +
         VCompany INDI ICompany
   ENDIF
   BREAK

--Move backward the specified number of records
CASE 'Jump Backward "n"'
   DIALOG 'How many to jump backward?' VNum=4 VEndKey 1
   SET VAR VMinus = (INT(.VNum) * -1)

   --R:BASE counts from the current cursor position
   FETCH RELATIVE .vminus FROM C1 INTO +
      VCustId INDI ICustId, VLastname INDI ILastname, +
      VCompany INDI ICompany

   --If the number of records to jump past takes you beyond the
   first
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
   --record, the first record is retrieved
   IF SQLCODE = 100 THEN
      FETCH FIRST FROM C1 INTO +
         VCustId INDI ICustId, VLastname INDI ILastname, +
         VCompany INDI ICompany
   ENDIF
   BREAK

--Jump to the last record Next customer from the last record returns
end-of-data
CASE 'Last Customer'
   FETCH LAST FROM C1 INTO +
      VCustId INDI ICustId, VLastname INDI ILastname, +
      VCompany INDI ICompany
   BREAK

--Jump to the first record Prior customer from the first record
returns end-of-data
CASE 'First Customer'
   FETCH FIRST FROM C1 INTO +
      VCustId INDI ICustId, VLastname INDI ILastname, +
      VCompany INDI ICompany
   BREAK

--Prompt for the last name to find
CASE 'Search by Last Name'
   SET VAR vsearch = NULL
   DIALOG 'Enter the last name to find' +
      VSearch VEndKey 1
   IF VEndKey = '[Esc]' THEN
      BREAK
   ENDIF

   WHILE #PI <> 0.0 THEN

      --Search forward for a matching record
      FETCH NEXT FROM c1 INTO +
         VCustID INDI ICustId, VLastname INDI ILastname, +
         VCompany INDI ICompany

      --If a match is found, the row is displayed and the cursor
      repositioned
      --at that row
      IF VLastname CONTAINS .VSearch THEN
         BREAK
      ENDIF

      --If no match was found, the search can be continued from the
      first row.
      IF SQLCODE = 100 THEN
         DIALOG 'No match found. Continue search from beginning?' +
            VResp VEndKey YES
         IF VEndKey = '[Esc]' THEN
            BREAK
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
            ENDIF

            IF VResp = 'YES' THEN
               FETCH FIRST FROM c1 INTO +
                  VCustID INDI ICustId, VLastname INDI ILastname, +
                  VCompany INDI ICompany
               IF VLastname CONTAINS .VSearch THEN
                  BREAK
               ENDIF
            ELSE
               --If the search is not continued, the last row is
               retrieved
               FETCH LAST FROM c1 INTO +
                  VCustID INDI ICustId, VLastname INDI ILastname, +
                  VCompany INDI ICompany
               BREAK
            ENDIF
         ENDIF
      ENDWHILE
      BREAK

   --Prompt for the company name to find
   CASE 'Search by Company'
      SET VAR VSearch = NULL
      DIALOG 'Enter the company to find' +
         VSearch VEndKey 1
      IF VEndKey = '[Esc]' THEN
         BREAK
      ENDIF

      --Search forward for a matching company record If a match is
      found,
      --the row is displayed and the cursor repositioned at that row
      WHILE #PI <> 0.0 THEN
         FETCH NEXT FROM c1 INTO +
            VCustID INDI ICustId, VLastname INDI ILastname, +
            VCompany INDI ICompany
         IF VCompany CONTAINS .VSearch THEN
            BREAK
         ENDIF

         --If no match was found, the search can be continued from the
         first row.
         IF SQLCODE = 100 THEN
            DIALOG 'No match found. Continue search from beginning?' +
               VResp VEndKey YES
            IF VEndKey = '[Esc]' THEN
               BREAK
            ENDIF
               IF VResp = 'YES' THEN
               FETCH FIRST FROM c1 INTO +
                  VCustID INDI ICustId, VLastname INDI ILastname, +
                  VCompany INDI ICompany
               IF VCompany CONTAINS .VSearch THEN
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
                  BREAK
               ENDIF
            ELSE
               --If the search is not continued, the last row is
               retrieved.
               FETCH LAST FROM c1 INTO +
                  VCustID INDI ICustId, VLastname INDI ILastname, +
                  VCompany INDI ICompany
               BREAK
            ENDIF
         ENDIF
      ENDWHILE
      BREAK
   ENDSW
ENDWHILE
DROP CURSOR C1
RETURN
```

1.7.6 Optimizing Cursors

DECLARE CURSOR is not always the fastest way to accomplish a task, particularly an UPDATE or an INSERT. If you can replace your DECLARE CURSOR routine with a single SQL command, you will dramatically improve performance. However, some tasks require a DECLARE CURSOR.

Let the cursor do the work
To improve the performance of a DECLARE CURSOR routine, do as much work in the DECLARE CURSOR as possible. This is the single most important factor in improving cursor performance. Do whatever work can be done in the SELECT command part of the DECLARE CURSOR - select as many columns of data as possible and also do calculations there if you can. The DECLARE CURSOR does the operation only once; inside the WHILE loop, the command is repeated for each row that is stepped through.

To do actions for unique rows only, use SELECT DISTINCT in the cursor definition instead of adding code to your WHILE loop to test the row values to see if they are the same or different. Use the SELECT functions to sum, average, count and so on in the cursor definition instead of for each row in the WHILE loop. Select as many columns as possible in the DECLARE CURSOR rather than retrieve the data each row in the WHILE loop.

The fewer commands repeated in the WHILE loop, the faster your DECLARE CURSOR will run. Remember that each command in the WHILE loop is repeated for each row retrieved by the DECLARE CURSOR. Use optimized variables in the WHILE loop -initialize each variable outside the WHILE loop, and do not change the data type of variables in the loop.

Following are two examples showing progressive changes made to a DECLARE CURSOR routine to improve performance.

Example 1 - posting. The task is to sum the extended price column in the transaction detail, Transdetail, table for each transaction ID, then update the transaction header, Transmaster, table with the sum. An initial approach is to declare a cursor on the header table, then step through all matching rows in the detail table. After all the matching detail rows have been processed, the header table is updated.

```
*(POST1.RMD -- the worst case)
-- nested declare cursors
-- strictly linear programming
SET VAR vtotal CURR
DECLARE c1 CURSOR FOR SELECT transid, netamount +
   FROM transmaster
OPEN c1
FETCH c1 INTO vtransid INDI vind1, vnetamount INDI vind2
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
WHILE SQLCODE <> 100 THEN
   DECLARE c2 CURSOR FOR SELECT extprice +
      FROM transdetail WHERE transid = .vtransid
   OPEN c2
   FETCH c2 INTO vprice INDI vind3
   WHILE SQLCODE <> 100 THEN
      SET VAR vtotal = (.vtotal + .vprice)
      FETCH c2 INTO vprice INDI vind3
   ENDWHILE
   DROP CURSOR c2
   UPDATE transmaster SET netamount = .vtotal +
      WHERE CURRENT OF c1
   SET VAR vtotal = NULL
   FETCH c1 INTO vtransid INDI vind1, vnetamount INDI vind2
ENDWHILE
DROP CURSOR c1
```

We can speed up this code by following the recommended structure for nested cursors. If we move the second DECLARE CURSOR out of the WHILE loop and reset the cursor instead of dropping it, this command file will execute faster. However, the best way to improve this code is by removing the second DECLARE CURSOR altogether. We don't need to step through all the rows in the detail table - we can compute the sum with a single SELECT command.

```
*(POST2.RMD - a little bit better)
-- use the SELECT or COMPUTE command
-- to calculate the sum instead of a nested cursor
SET VAR vprice CURR = NULL
DECLARE c1 CURSOR FOR SELECT transid, netamount +
   FROM transmaster
OPEN c1
FETCH c1 INTO vtransid INDI vind1, vamount INDI vind2
WHILE SQLCODE <> 100 THEN
   SELECT SUM(extprice) INTO vprice +
      FROM transdetail WHERE transid = .vtransid
   -- if no matching rows in the Transdetail table,
   -- vprice is null
   IF vprice IS NOT NULL THEN
      UPDATE transmaster SET netamount = .vprice +
         WHERE CURRENT OF c1
   ENDIF
   SET VAR vprice = NULL
   FETCH c1 INTO vtransid INDI vind1, vamount INDI vind2
ENDWHILE
DROP CURSOR c1
```

This simple change reduced the number of commands in the program, which in turn improved performance. All the commands inside the WHILE loop still need to be executed for as many rows as are in the Transmaster table, however. The Transmaster table has fewer rows than the Transdetail table, so a valid assumption is to place the cursor on the Transmaster table to repeat the WHILE loop the fewest times.

However, if we place the cursor on the detail table instead of on the header table, the sum can be calculated directly in the DECLARE CURSOR. Because the command is grouped by the transaction ID, the same number of rows is retrieved by the cursor. The only commands to repeat in the WHILE loop are the UPDATE and the FETCH to get the next row. At first this might seem backwards, but computing the sum in the DECLARE CURSOR is much faster.

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
*(POST3.RMD - better yet)
-- declare the cursor on the detail table and
-- do the sum directly in the cursor definition
DECLARE c1 CURSOR FOR SELECT transid, SUM(extprice) +
   FROM transdetail GROUP BY transid
OPEN c1
FETCH c1 INTO vtransid INDI vind1, vprice INDI vind2
WHILE SQLCODE <> 100 THEN
   -- this is a non-updatable cursor so an explicit
   -- WHERE clause is used
   UPDATE transmaster SET netamount = .vprice +
      WHERE transid = .vtransid
   FETCH c1 INTO vtransid INDI vind1, vprice INDI vind2
ENDWHILE
DROP CURSOR c1
```

The number of commands has been reduced by over half from the first program, and performance by more than that. The multi-table update command is actually the fastest way to accomplish this task.

```
*(POST4.RMD - do a multi-table update if you can)
-- multi table update command, a view is used
-- to first calculate the sum and create a
-- one-one relationship
DROP VIEW v_trans
CREATE VIEW v_trans (transid, amount) AS +
   SELECT transid, SUM(extprice) +
   FROM transdetail GROUP BY transid
UPDATE transmaster SET netamount = amount +
   FROM transmaster ,v_trans t2 +
   WHERE transmaster.transid = t2.transid
```

Example 2 - a quick report. The task here is to create a quick report of companies from the Customer table and their corresponding contact names from the Contact table. Using nested cursors makes printing the company information once followed by the many rows of contact information easier.

```
*(CUSTREP1.RMD - the worst case)
-- nested cursors are used with the declare for
-- the second cursor inside the while loop of
-- the first cursor. Also, the data is retrieved
-- with a SELECT command instead of in the
-- cursor definition

-- Dropping a cursor before you declare it is a
-- technique used to guarantee that the cursor does
-- not exist in memory. The DROP CURSOR normally
-- returns an error message, so check to verify it
-- exists before dropping it.
SET VAR vCheckCursor1 INTEGER = (CHKCUR('c1'))
IF vCheckCursor1 = 1 THEN
  DROP CURSOR c1
ENDIF
SET VAR vCheckCursor2 INTEGER = (CHKCUR('c2'))
IF vCheckCursor2 = 1 THEN
  DROP CURSOR c2
ENDIF
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
-- Only the unique row identifier is specified in
-- the cursor definition
DECLARE c1 CURSOR FOR SELECT custid FROM customer +
   ORDER BY custid
OPEN c1
FETCH c1 INTO vcustid INDI ind1
WHILE SQLCODE <> 100 THEN

   -- Retrieve and display the rest of the
   -- data for a customer
   SELECT company, custaddress, custcity, +
      coststate, custzip, custphone INTO +
      vcompany INDI vi1, vaddress INDI vi2, +
      vcity INDI vi3, vstate INDI vi4, +
      vzipcode INDI vi5, vphone INDI vi6 +
      FROM customer WHERE custid = .vcustid
   SET VAR vcsz = (.vcity + ',' & .vstate & .vzipcode)
   WRITE .vcustid, .vcompany
   WRITE .vaddress
   WRITE .vcsz

   -- Declare a cursor to identify matching contact rows
   DECLARE c2 CURSOR FOR SELECT contfname, contlname +
      FROM contact WHERE custid = .vcustid
   OPEN c2
   FETCH c2 INTO vfname INDI i1, vlname INDI i2
   WHILE SQLCODE <> 100 THEN
      SET VAR vfullname = (.vfname & .vlname)
      WRITE .vfullname
      FETCH c2 INTO vfname INDI i1, vlname INDI i2
   ENDWHILE
   DROP CURSOR c2
   FETCH c1 INTO vcustid INDI ind1
ENDWHILE
DROP CURSOR c1
```

The next code segment shows the recommended structure for nested cursors. The second DECLARE CURSOR is moved to the top of the program, and the second cursor is opened and closed, not declared and dropped. Just this simple change improves performance.

```
*(CUSTREP2.RMD - move cursor out of WHILE loop)
SET VAR vCheckCursor1 INTEGER = (CHKCUR('c1'))
IF vCheckCursor1 = 1 THEN
  DROP CURSOR c1
ENDIF
SET VAR vCheckCursor2 INTEGER = (CHKCUR('c2'))
IF vCheckCursor2 = 1 THEN
  DROP CURSOR c2
ENDIF

SET VAR vcustid INTEGER
DECLARE c1 CURSOR FOR SELECT custid +
   FROM customer ORDER BY custid
DECLARE c2 CURSOR FOR SELECT contfname, contlname +
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
   FROM contact WHERE custid = .vcustid

-- Get the first row of data for a customer
OPEN c1
FETCH c1 INTO vcustid INDI ind1
WHILE SQLCODE <> 100 THEN

   -- Retrieve and display the rest of the
   -- data for a customer
   SELECT company, custaddress, custcity, +
      coststate, custzip, custphone INTO +
      vcompany INDI vi1, vaddress INDI vi2, +
      vcity INDI vi3, vstate INDI vi4, +
      vzipcode INDI vi5, vphone INDI vi6 +
      FROM customer WHERE custid = .vcustid
   SET VAR vcsz = (.vcity + ',' & .vstate & .vzipcode)
   WRITE .vcustid, .vcompany
   WRITE .vaddress
   WRITE .vcsz

   -- Open cursor c2, retrieve and display
   -- the matching contact data
   OPEN c2
   FETCH c2 INTO vfname INDI i1, vlname INDI i2
   WHILE SQLCODE <> 100 THEN
      SET VAR vfullname = (.vfname & .vlname)
      WRITE .vfullname
      FETCH c2 INTO vfname INDI i1, vlname INDI i2
   ENDWHILE

   -- Close cursor c2 and get the next row of
   -- customer data
   CLOSE c2
   FETCH c1 INTO vcustid INDI ind1
ENDWHILE
DROP CURSOR c1
DROP CURSOR c2
```

Moving the data retrieval to the DECLARE CURSOR command instead of using a separate SELECT command again improves performance.

```
*(CUSTREP3.RMD)
--retrieve data through DECLARE CURSOR
SET VAR vCheckCursor1 INTEGER = (CHKCUR('c1'))
IF vCheckCursor1 = 1 THEN
  DROP CURSOR c1
ENDIF
SET VAR vCheckCursor2 INTEGER = (CHKCUR('c2'))
IF vCheckCursor2 = 1 THEN
  DROP CURSOR c2
ENDIF

-- retrieve all the data through the DECLARE CURSOR
-- command instead of SELECT
SET VAR vcustid INTEGER
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
DECLARE c1 CURSOR FOR SELECT custid, company, +
   custaddress, custcity ,coststate, custzip, +
   custphone FROM customer ORDER BY custid
DECLARE c2 CURSOR FOR SELECT contfname, contlname +
   FROM contact WHERE custid = .vcustid
OPEN c1

-- Get the first row of customer data
FETCH c1 INTO vcustid INDI ind1, vcompany INDI ind2,+
   vaddress INDI ind3, vcity INDI ind4, vstate INDI ind5, +
   vzip INDI ind6, vphone INDI ind7
WHILE SQLCODE <> 100 THEN

   -- Display the customer data and open cursor c2 to
   -- retrieve the matching contact data
   SET VAR vcsz = (.vcity + ',' & .vstate & .vzipcode)
   WRITE .vcustid, .vcompany
   WRITE .vaddress
   WRITE .vcsz
   OPEN c2
   FETCH c2 INTO vfname INDI i1, vlname INDI i2
   WHILE SQLCODE <> 100 THEN
      SET VAR vfullname = (.vfname & .vlname)
      WRITE .vfullname
      FETCH c2 INTO vfname INDI i1, vlname INDI i2
   ENDWHILE

   -- Close cursor c2 and get the next row of
   -- customer data
   CLOSE c2
   FETCH c1 INTO vcustid INDI ind1, vcompany INDI ind2,+
      vaddress INDI ind3, vcity INDI ind4, vstate INDI ind5, +
      vzip INDI ind6, vphone INDI ind7
ENDWHILE
DROP CURSOR c1
DROP CURSOR c2
```

Another small change also improves performance - instead of using SET VAR commands within the WHILE loops to concatenate city, state and zipcode together, and first and last name together, the concatenation operation can be done in the DECLARE CURSOR command. The concatenation in the DECLARE CURSOR reduces the number of commands that are repeated for each row and moves the work to the DECLARE CURSOR command.

```
*(CUSTREP4.RMD add the concatenation to the DECLARE CURSOR)
SET VAR vCheckCursor1 INTEGER = (CHKCUR('c1'))
IF vCheckCursor1 = 1 THEN
  DROP CURSOR c1
ENDIF
SET VAR vCheckCursor2 INTEGER = (CHKCUR('c2'))
IF vCheckCursor2 = 1 THEN
  DROP CURSOR c2
ENDIF

SET VAR vcustid INTEGER
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
-- Replace SET VAR commands with expressions in
-- the DECLARE CURSOR
DECLARE c1 CURSOR FOR SELECT custid, company, +
custaddress, (custcity + ',' & custstate & custzip), +
   custphone FROM customer ORDER BY custid
DECLARE c2 CURSOR FOR SELECT (contfname & contlname) +
   FROM contact WHERE custid = .vcustid
OPEN c1

-- Retrieve and display the customer data
FETCH c1 INTO vcustid INDI ind1, vcompany INDI ind2, +
   vaddress INDI ind3, vcsz INDI ind4, vphone INDI ind5
WHILE SQLCODE <> 100 THEN
   WRITE .vcustid, .vcompany
   WRITE .vaddress
   WRITE .vcsz

   -- Retrieve and display the contact data
   OPEN c2
   FETCH c2 INTO vfullname INDI i1
   WHILE SQLCODE <> 100 THEN
      WRITE .vfullname
      FETCH c2 INTO vfullname INDI i1
   ENDWHILE

   -- Close cursor c2 and get the next row of
   -- customer data
   CLOSE c2
   FETCH c1 INTO vcustid INDI ind1, vcompany INDI ind2, +
      vaddress INDI ind3, vcsz INDI ind4, vphone INDI ind5
ENDWHILE
DROP CURSOR c1
DROP CURSOR c2
```

The final change to improve performance is to use the RESET option on the OPEN c2 command instead of CLOSE c2. Overall, we have improved performance on this small set of rows by a full second. On a larger data set you can expect to see a greater performance improvement.

```
*(CUSTREP5.RMD)
--reset cursor 2 instead of close and open

SET VAR vCheckCursor1 INTEGER = (CHKCUR('c1'))
IF vCheckCursor1 = 1 THEN
  DROP CURSOR c1
ENDIF
SET VAR vCheckCursor2 INTEGER = (CHKCUR('c2'))
IF vCheckCursor2 = 1 THEN
  DROP CURSOR c2
ENDIF

SET VAR vcustid INTEGER
DECLARE c1 CURSOR FOR SELECT custid, company, +
   custaddress, (custcity + ',' & custstate & custzip), +
   custphone FROM customer ORDER BY custid
DECLARE c2 CURSOR FOR SELECT (contfname & contlname) +
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
   FROM contact WHERE custid = .vcustid
OPEN c1
FETCH c1 INTO vcustid INDI ind1, vcompany INDI ind2, +
   vaddress INDI ind3, vcsz INDI ind4, vphone INDI ind5
WHILE SQLCODE <> 100 THEN
   WRITE .vcustid, .vcompany
   WRITE .vaddress
   WRITE .vcsz

   -- Open cursor c2 with the RESET option,
   -- no CLOSE command is needed
   OPEN c2 RESET
   FETCH c2 INTO vfullname INDI i1
   WHILE SQLCODE <> 100 THEN
      WRITE .vfullname
      FETCH c2 INTO vfullname INDI i1
   ENDWHILE
   FETCH c1 INTO vcustid INDI ind1, vcompany INDI ind2, +
      vaddress INDI ind3, vcsz INDI ind4, vphone INDI ind5
ENDWHILE
DROP CURSOR c1
DROP CURSOR c2
```

As you can see from the above examples, maximizing the work of the DECLARE CURSOR command provides significant performance improvements. The changes were small and they didn't involve a lot of time or programming effort, but these changes did result in definite performance benefits.

Customize the environment
In addition to optimizing your programming code, you can improve cursor performance by optimizing the environment. Obviously, code runs faster on a newer computer. Outside of upgrading your hardware, however, certain R:BASE environment settings can be used to improve performance. These settings generally improve overall performance as well as cursor performance.

Look at the EXPLAIN.DAT output file generated by the MICRORIM_EXPLAIN variable to see the cursor query optimization. The OPEN command actually executes the query. Each query executed in your program puts an entry in EXPLAIN.DAT; for example, SELECT or UPDATE commands in the WHILE loop are reflected. You might also see a query reference to the SYS_RULES table, which is used for multi-user locking control.

By using EXPLAIN.DAT, you can easily see why using the RESET option on OPEN is faster. Normally, each OPEN redoes the query. When RESET is used, the query is only optimized once.

The EXPLAIN.DAT entries for the last two command files (CUSTREP4.RMD and CUSTREP5.RMD) from **Example 2** are shown here. The first entry shows nested cursors using the OPEN and CLOSE commands. The second entry shows using the RESET option on OPEN.

Cursor c1 on the Customer table is accessed sequentially, all rows in the table are retrieved, and no WHERE clause is used. If an indexed WHERE clause was used, EXPLAIN.DAT would show the index used. The second cursor on the Contact table does use an indexed WHERE clause to define the query. This query is redone each time the cursor is opened with a different vcustid value.

```
SortStrategy = DB_TAG  (internal=1)

SelectCost=1.  (OptimizationTime=0ms)
  Customer Sequential

SelectCost=2.904827e-002  (OptimizationTime=0ms)
  Contact (ColumnName=CustID,Type=F) Random Dup=1.296296 Adj=0.9714286
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
SelectCost=2.904827e-002  (OptimizationTime=0ms)
  Contact (ColumnName=CustID,Type=F) Random Dup=1.296296 Adj=0.9714286

SelectCost=2.904827e-002  (OptimizationTime=0ms)
  Contact (ColumnName=CustID,Type=F) Random Dup=1.296296 Adj=0.9714286
....

SelectCost=2.904827e-002  (OptimizationTime=0ms)
  Contact (ColumnName=CustID,Type=F) Random Dup=1.296296 Adj=0.9714286

SelectCost=2.904827e-002  (OptimizationTime=0ms)
  Contact (ColumnName=CustID,Type=F) Random Dup=1.296296 Adj=0.9714286

SelectCost=2.904827e-002  (OptimizationTime=0ms)
  Contact (ColumnName=CustID,Type=F) Random Dup=1.296296 Adj=0.9714286

SelectCost=1.  (OptimizationTime=0ms)
  SYS_RULES Sequential
```

The following EXPLAIN.DAT entry uses OPEN c2 RESET. The same query is used each time cursor c2 is accessed. The query does not need to be reoptimized each time the cursor is opened.

```
SortStrategy = DB_TAG  (internal=1)

SelectCost=1.  (OptimizationTime=0ms)
  Customer Sequential

SelectCost=2.904827e-002  (OptimizationTime=0ms)
  Contact (ColumnName=CustID,Type=F) Random Dup=1.296296 Adj=0.9714286

SelectCost=1.  (OptimizationTime=0ms)
  SYS_RULES Sequential
```

For additional information on using the MICRORIM_EXPLAIN variable to see the cursor query optimization, refer to the "Environment Optimization" chapter within the Reference Index of the R:BASE Help.

1.7.7 Questions & Answers

Q. When should I use a cursor?
A. Use a cursor when it seems like the best way to get a task done. There are no rules or standards to say when you should use a cursor and when you shouldn't. Often the logic behind a cursor is easier to understand than the logic behind a complex SELECT or UPDATE command that works across a group of rows. Many programmers have replaced DECLARE CURSOR routines with a single INSERT, UPDATE or DELETE command, most often for performance reasons, but not all cursors can be replaced with a single SQL command.

Deciding to use a cursor will depend on your level of programming expertise and understanding of the task to be accomplished. First get the program to work; once it works, look at ways to make the program run more efficiently and faster.

Q. How do I make a cursor faster?

A. Using a DECLARE CURSOR is slower than using just a single SQL command working across a group of rows, but some tasks just can't be done without using a cursor. You can use certain techniques to maximize the performance of DECLARE CURSOR routines. However, just like deciding when to use a cursor, there are no rules or standards about improving the performance of a cursor.

Copyright © 1982-2024 R:BASE Technologies, Inc.

One of the best ways to make a cursor faster is to move as much of the work as possible into the DECLARE CURSOR command itself. Let the cursor select as many columns as possible. If you are doing calculations for each row, see if you can use one of the SELECT functions with the GROUP BY option.

For additional suggestions to improve cursor performance, see the Optimizing Cursors chapter in this document.

Q. Should I use WHERE CURRENT OF or an explicit WHERE clause?
A. In terms of performance, there is very little difference between the two options. Not all cursors can be used with the WHERE CURRENT OF syntax. Getting the most out of your DECLARE CURSOR statement is more important in terms of performance than making your cursor an updatable cursor.

Q. I'm trying to UPDATE data using WHERE CURRENT OF and I get a syntax error. I have checked and double checked the syntax, and it is fine.
A. You get this error when you have a non-updatable cursor. A non-updatable cursor does not support use of WHERE CURRENT OF. Use an explicit WHERE clause to update the table instead of WHERE CURRENT OF.

Q. What is a non-updatable cursor?
A. A cursor knows what data to retrieve based on the SELECT statement that is part of the DECLARE CURSOR command. Like a regular SELECT command, the SELECT that is part of the DECLARE CURSOR can retrieve data from multiple tables or use a GROUP BY. It has all the features of the regular SELECT. However, only a single table SELECT with no GROUP BY is updatable; this option is the only one that guarantees the cursor is pointing to a single row in a table. If the cursor can't point back to and identify a single row, it doesn't know what to update.

Q. Is it faster to retrieve data inside my WHILE loop using the SET VAR command or the SELECT...INTO command?
A. It's just a little bit faster to retrieve additional data using a SET VAR command instead of the SELECT...INTO command. The SELECT has more overhead. The fastest way to retrieve column data into variables, however, is to retrieve whatever columns possible through the DECLARE CURSOR command. That method can be almost twice as fast as using either SET VAR or SELECT...INTO.

Q. My WHILE loop never ends. It just keeps repeating the last row.
A. FETCH, which sets SQLCODE, should be the last command in the WHILE loop. When no more data is available, SQLCODE is set to 100. If FETCH is the last command in the WHILE loop, the next command executed is the WHILE statement, which tests the current value of SQLCODE. Other SQL commands placed after the FETCH and before the ENDWHILE might reset SQLCODE to a value other than 100.

Also, if your WHILE condition is not SQLCODE <> 100, make sure you are checking the condition correctly. If the WHILE loop doesn't exit, the WHILE condition is never false. Use TRACE and set up watch variables to see what is happening with your variable values.

Q. Why won't WHENEVER work with DECLARE CURSOR?
A. WHENEVER is an SQL error trap command that executes a GOTO whenever the data not found situation (SQLCODE = 100) occurs. At first glance, WHENEVER seems ideal for use with a DECLARE CURSOR. However, if your DECLARE CURSOR routine uses any other SQL commands that can return a "data not found" error, such as SELECT, INSERT or UPDATE, the WHENEVER immediately exits the DECLARE CURSOR WHILE loop even though all the data has not been processed. The R:BASE error "No rows exist or satisfy the WHERE clause" is a "data not found" error and sets SQLCODE to 100.

Q. I use DECLARE CURSOR to find out if a row exists in a table. Is there a way to do this check faster?
A. If you only want to see if a row exists in a table, don't use DECLARE CURSOR. The DECLARE CURSOR command by itself doesn't check this. You need to OPEN the cursor and FETCH before you know if a row has been found. Instead use the SELECT command; SELECT INTO a variable and test the variable value, or test SQLCODE immediately after the SELECT command. If no row is found, SQLCODE is set to 100. Using just the SELECT command is much faster than using the DECLARE CURSOR.

Q. My DECLARE CURSOR command is giving me a syntax error. Is there an easy way to check the syntax?
A. First make sure the cursor name is in the correct place in the command. A common error is to use DECLARE CURSOR c1 instead of DECLARE c1 CURSOR. The SELECT part of the DECLARE CURSOR command can get quite complex, particularly when more than one table is involved. Test the SELECT part of the DECLARE CURSOR command at the R> Prompt, which executes just like a regular SELECT

Copyright © 1982-2024 R:BASE Technologies, Inc.

command. You can test and debug the SELECT part of your DECLARE CURSOR before putting it into the DECLARE CURSOR structure.

1.8 Data Types

These data types can be specified within the R:BASE command syntax using the SET VARIABLE command.

BIGINT
- Holds a 64-bit integer value
- Offers a range of ±999,999,999,999,999,999
- Delimiters (such as commas) cannot be used in entry
- No length is needed

BIGNUM
- Holds decimal numbers whose precision and scale can be set
- When specifying BIGNUM, specify a precision (the total number of digits) from 1 to 38 (default 18) and a scale (the number of decimal places) from zero to any positive integer up to the precision value (default 0)
- R:BASE reserves a minimum of forty bytes of internal storage
- BIGNUM numbers are stored as DECIMAL

BIT
- Holds binary data
- The default length is 1 bit
- The fixed length is 1 to 1,500 bytes

BIT VARYING
Maps to VARBIT

BITNOTE
- Holds binary data
- No length is needed
- The variable length is 0 to 4,088 bytes of binary data

BOOLEAN
- Hold true/false values
- Internally stored as 0 for false and 1 for true
- Accepted values for false include: 0, false, 'false', f
- Accepted values for true include: 1, true, 'true', t

Notes:

- For multilingual applications, the values of 0 and 1 are recommended.
- The BOOLEAN operating condition setting specifies that constants (e.g. TRUE, FALSE) in expressions will be treated as type BOOLEAN values. The default is OFF.

BSTR
- Holds binary string data
- String data type that is used by COM (Component Object Model), Automation, and Interop functions
- Used to support Unicode in table data
- Composite data type that consists of a length prefix, a data string, and a terminator

Length Prefix
- Consists of a four-byte integer
- Occurs immediately before the first character of the data string
- Contains the number of bytes in the following data string
- Does not include the terminator

Data String
- Consists of a string of Unicode characters (wide or double-byte characters)
- May contain multiple embedded null characters

Terminator

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Consists of two null characters (0x00)

CHAR VARYING
Maps to VARCHAR

CHARACTER
Maps to TEXT

CURRENCY
- Holds monetary values of up to 23 digits represented in the currency format, established using SET CURRENCY
- Dollar amounts are in the range ±$99,999,999,999,999.99
- Commas or the current delimiter can be used. If no decimal point is included, .00 is assumed
- Data is stored as two long integer values, reserving four bytes of internal storage
- The negative currency format with parenthesis around the negative value e.g. ($500.00), is not recognized

DATE
- Holds dates in a 1- to 30-character format based on month, day, and year as established using SET DATE
- The minimum and default format to display month, day, and year is MM/DD/YY
- The allowable date range is January 1, 3999 BC to December 31, 9999 AD
- R:BASE reserves four bytes of internal storage

DATETIME
- A concatenation of the DATE and TIME data types, resulting in a sequence and display format as set by DATE and TIME
- DATETIME cannot be SET directly, but does permit extraction of its value by the DATETIME functions into a DATETIME variable
- For example: SET VAR vdatetime = (DATETIME(06/12/93, 12:15:30.123)), results in vdatetime = '06/12/93', 12:15:30.123'
- For identification purposes, DATETIME values are automatically stamped into R:BASE databases, also known as the *timestamp*
- DATETIME occupies 8 bytes of internal storage
- The time portion of the value does not have to be specified in a DATETIME data type. If omitted, it defaults to 0:0:0

DECIMAL
Maps to NUMERIC

DOUBLE
- Holds double-precision real numbers in the range ±1.7E308 with a precision of up to 15 digits
- DOUBLE numbers longer than 15 digits are stored as scientific notation
- R:BASE reserves eight bytes of internal storage
- Because DOUBLE numbers are stored in a binary form, the displayed value may not be the stored value
- Calculations are performed on the stored values
- Among numeric data types, DOUBLE provides the greatest range of values

GUID
- Holds a 128-bit value
- Binary global unique identifier, which is represented as a 32 hexadecimal digits
- Listed as a group of 8 digits, followed by three groups of 4 digits, followed by a group of 12 digits, for a total of 32 digits, separated by hyphens
- As the GUID data type is a binary unique value, it will increase retrieval of data from tables for indexed columns

An example of a GUID value is: **8C20005C-0E2A-47E0-B2BE-57E67961628B**

INTEGER
- Holds whole numbers in the range of ±1,999,999,999
- Delimiters (such as commas) cannot be used in entry
- R:BASE reserves four bytes of internal storage space

LONG VARBINARY

Copyright © 1982-2024 R:BASE Technologies, Inc.

Maps to VARBIT

LONG VARBIT
Maps to VARBIT

LONG VARCHAR
Maps to VARCHAR

NOTE
- Holds alphanumeric data
- The default length is 0, where the length is determined by the data
- Holds variable length text of up to 10,000 characters
- Maximum length of a NOTE column can be set
- Indexes and constraints are allowed on NOTE data types
- R:BASE reserves a minimum of four bytes of internal storage space

NUMERIC
- Holds decimal numbers whose precision and scale can be set
- When specifying NUMERIC, specify a precision (the total number of digits) from 1 to 15 (default 9) and a scale (the number of decimal places) from zero to any positive integer up to the precision value (default 0)
- R:BASE reserves a minimum of eight bytes of internal storage
- NUMERIC numbers are stored as DOUBLE

REAL
- Holds real number amounts in the range of ±1E38 with six-digit accuracy
- Real numbers with up to seven digits are displayed as decimal numbers; for example, 321.414
- Real numbers with more than seven digits are represented in scientific notation; for example, 9.8E32
- R:BASE reserves four bytes of internal storage space
- REAL numbers are stored in a binary form; therefore, the displayed value may not be the actual stored value
- Calculations are performed on the stored values

SMALLINT
- Holds a 16-bit integer value
- Offers a range of ±32767
- Delimiters (such as commas) cannot be used in entry
- No length is needed

TEXT
- Holds alphanumeric data
- The default length is eight characters
- The maximum is 1,500 characters
- Maximum length of a TEXT column can be set
- R:BASE reserves a minimum of four bytes of internal storage space

TIME
- Holds time values in a 1- to 20-character format based on hours, minutes, and seconds, established using SET TIME
- The minimum format to display hours, minutes, and seconds is HH:MM:SS
- TIME can be specified up to thousandths of a second
- Time can be displayed or entered as a 12- or 24-hour clock
- R:BASE reserves four bytes of internal storage

VARBINARY
Maps to VARBIT

VARBIT
- Holds binary data
- No length is needed
- Is ideal for storing external files, like images, PDF files, etc.

VARCHAR
- Holds alphanumeric data
- No length is needed

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Is ideal for storing large text data

WIDENOTE
- Holds Unicode data
- The default length is 0, where the length is determined by the data
- Holds variable length text of up to 4,092 characters
- Maximum length of a WIDENOTE column can be set
- Indexes and constraints are allowed on WIDENOTE data types
- R:BASE reserves a minimum of four bytes of internal storage space, with 2 bytes per character

WIDETEXT
- Holds Unicode data
- The default length is eight characters
- The maximum is 1,500 characters
- Maximum length of a WIDETEXT column can be set
- R:BASE reserves a minimum of four bytes of internal storage space, with 2 bytes per character

1.9 Database Conversion

1.9.1 Preparation

When preparing to convert a database and use those files in R:BASE 11, it is important that you follow the proper steps to convert that database. Please review the following steps and recommendations to ensure you convert your legacy database(s) successfully.

1.9.1.1 Make a Database Backup

Before you begin, make a backup copy of your database. You should place the database on a storage medium that is different than the one which your original files are already stored upon. For example, if you are using a hard disk to store the data, you can back up your database to an external hard drive, a flash drive, a tape drive, or a CD-ROM. With these storage options, you can also copy the backup files back to your hard drive.

If users are still entering data into the live database, while the conversion is taking place, use a different database copy than the one in use. When the conversion is complete, the live data being added to can always be loaded into the converted database at a later time. Make sure no users are connected to the database when the copy is made.

During the conversion process, it is also recommended to make periodic backups. This will prevent you from starting from the beginning in case any problems occur.

1.9.1.2 Review Table and Column Names

You must review all table and column names to be sure that no invalid names are defined and no reserved words are used.

In the current version, a 1-18 character alpha-numeric name must be specified for a column. Spaces are NOT permitted. Valid names must start with an alpha character and can include the following symbols:

- Letters (A-Z)
- Numbers (0-9)
- _ (underscore)
- # (pound sign) *
- $ (dollar sign) *
- % (percent sign) *

*__Note:__ For ODBC compliance, it is NOT recommended to use the #, $, or % even though R:BASE permits it.

Review your table and column names that may begin with a number. Check for any characters that are not supported like a question mark "?" or a greater than character ">".

Copyright © 1982-2024 R:BASE Technologies, Inc.

Also review your table and column names to verify that no reserved words are used. These words, commands, keywords, and other names are to be used exclusively by the R:BASE database. Names used by System Tables, System Views and System Columns are also reserved words. As a rule, all words beginning with "**SYS**" are reserved.

Do not use reserved words or any shorter forms of them as names for columns, tables, or views. As a rule, if the word is a reserved word, R:BASE will flag it. R:BASE will not allow you to use a reserved word as a column or table name, but this MAY NOT always be the case. For example in the table designer, R:BASE may not warn you about **REF**, short for **REFERENCES**, but **REF** will not be allowed in a command file. If a particular column or table is giving you problems, please check out the list below and consider all shortened versions of the words listed here.

The following is a list of all reserved words.

#DATE	LE
#PI	LEAD
#TIME	LIKE
ADA	LIMIT
ADD	LISTREL
ALL	LT
AND	MAX
ANY	MAXIMUM
AS	MIN
ASC	MINIMUM
ASCII	MODULE
AT	MPW
AUTHORIZE	NE
AUTONUMBER	NOCHECK
AVERAGE	NOFILL
AVG	NONE
BEGINS	NONUM
BETWEEN	NOT
BLINKING	NULL
BOTH	NUM
BY	OF
CASCADE	OFF
CHARACTER	ON
CHECK	OPTION
CMDHIST	OR
COLUMNS	ORDER
CONSTRAINT	OUTER
CONTAINS	OVER
CONTINUE	OWNER
COUNT	PARTITION
CURRENT	PLI
CURSORS	PRECISION
DATA	PRINTER
DBCONN	PRIVILEGE
DECIMAL	PROCEDURE
DESC	PUBLIC
DISTINCT	READ
DUPLICATE	REFERENCES
END	RPW
ENDC	SCHEMA
EQ	SCREEN
ESCAPE	SECTION
EXCEPT	SELECT
EXECUTE	SMALLINT
EXISTS	SOME
EXPLAIN	SORTED
FAILS	SOUNDS_LIKE
FILL	SQLCODE
FLOAT	SQLERROR
FOR	STATICVAR
FOUND	STDEV

Copyright © 1982-2024 R:BASE Technologies, Inc.

FROM	STRUCTURE
FULL	SUM
GE	TABLE
GO	TABLES
GOTO	TEMPORARY
GROUPED	TERMINAL
GT	TJOURNAL
HAVING	TO
IN	UNION
INDEX	UNIQUE
INDICATOR	USING
INTO	VALUES
IS	VARIANCE
KEY	VIEWS
KEYBOARD	WHERE
LAG	WITH
LANGUAGE	WORK
LAST	

Of the above, the following are new reserved words added with the release of R:BASE 11.

- CMDHIST
- STATICVAR
- EXPLAIN
- DBCONN
- TJOURNAL

If you are still using a legacy version of R:BASE, it is recommended that you perform these table and column name changes in that version. After making the table and column names changes, R:BASE will update the form, report, and label column objects with the appropriate new name. However, if any of the columns and tables are listed in the form, report, and label variable expressions, you must manually edit these expressions.

On paper, record any table and column name changes. Later in the conversion process when you're updating your command files, you will need to make these necessary name changes for the command files as well.

1.9.2 Converting the Database Files

1.9.2.1 Recognizing the Database Files

When locating and using the database name in the conversion instructions, be sure to remember that the "actual" database name for .RBS and .RBF files does not include the "1", "2", or "3", which is located before the file extension. For example, a R:BASE 3.1 database with the name SALES will have the

Copyright © 1982-2024 R:BASE Technologies, Inc.

database files SALES1.RBF, SALES2.RBF, and SALES3.RBF, but the database name is still just "SALES". In the steps below where the "dbname" is required in the command syntax, use only the "actual" database name.

To use this document you will need to know which version of R:BASE you are upgrading from. The table below lists the various R:BASE file formats. Once you find your R:BASE database version from the table below, follow the necessary conversion steps in the next section.

R:BASE Database File Formats

Version	File Format
R:BASE 4000 and R:BASE 5000	DBNAME1.RBS DBNAME2.RBS DBNAME3.RBS
R:BASE System V to R:BASE 4.0	DBNAME1.RBF DBNAME2.RBF DBNAME3.RBF
R:BASE 4.5 to R:BASE 5.0	DBNAME.RB1 DBNAME.RB2 DBNAME.RB3
R:BASE 5.1 to R:BASE 10.x	DBNAME.RB1 DBNAME.RB2 DBNAME.RB3 DBNAME.RB4
R:BASE Turbo V-8, R:BASE eXtreme 9.x (64), R:BASE 10.x Enterprise, and R:BASE 11	DBNAME.RX1 DBNAME.RX2 DBNAME.RX3 DBNAME.RX4

1.9.2.2 Conversion Steps

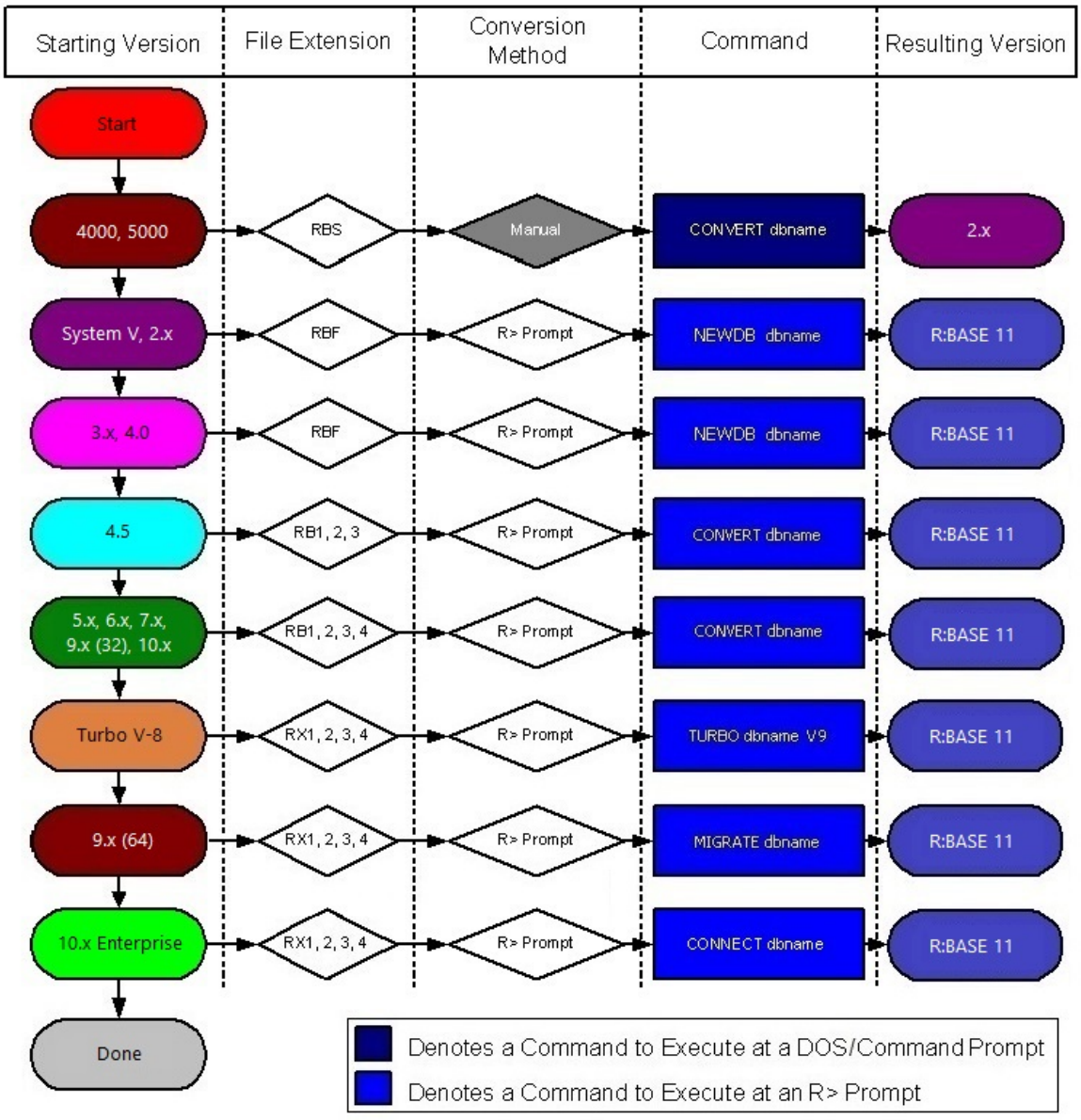
PLEASE REMEMBER TO BACK UP YOUR DATABASE!

The steps required to convert databases to work with R:BASE 11 for Windows are described in the following flow chart. Locate your current R:BASE database file "Starting Version" in the left column and complete the steps until you reach the bottom. If you are not sure of your database file version, refer to the Recognizing the Database Files chart.

In the sections that follow, proceed to the chapter that lists the version you are upgrading from. Your options include:

- Converting from R:BASE 4000/5000
- Converting from R:BASE System V to R:BASE 4.0
- Converting from R:BASE 4.5 through 6.5++
- Converting from R:BASE 7.x/9.x (32)/10.x
- Converting from R:BASE Turbo V-8
- Converting from R:BASE eXtreme 9.x (64)
- Converting from R:BASE X/X.5 Enterprise (10.x)

Copyright © 1982-2024 R:BASE Technologies, Inc.



1.9.2.2.1 Converting from R:BASE 4000/5000

If you have not done so already, please contact R:BASE Technologies to receive the legacy conversion utility for migrating R:BASE 4000/5000 databases. The legacy conversion utility is a 16-bit application file that will not open on 64-bit operating systems.

IMPORTANT: The instructions below require the steps be executed on a 32-bit operating system.

1. Create a new directory; perhaps something near your root directory on the local drive (example: "C:\DBTEMP").
2. Copy the contents of the "CONVERT" directory provided by R:BASE Technologies into the directory created in Step 1.
3. Copy your .RBS database files to the new directory.
4. Open a Command Prompt or DOS Prompt window and navigate to the new directory created above in Step 1.
5. At the Prompt, type: `CONVERT dbname`

Copyright © 1982-2024 R:BASE Technologies, Inc.

6. Repeat for all databases. In addition to the original .RBS database files, three new files will be created for each data that have the .RBF file extensions.
7. Exit the Command Prompt or DOS Prompt window. Use the EXIT command to exit.
8. Continue with the steps listed at Converting from R:BASE System V to R:BASE 4.0

1.9.2.2.2 Converting from R:BASE System V to R:BASE 4.0

At this point, you must have R:BASE 11 installed on your computer. Be sure to review the R:BASE 11 Getting Started Guide regarding the installation and set up for your R:BASE software. The Getting Started Guide is located in the R:BASE program directory.

If there is no OWNER or Password

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt type: `NEWDB DBNAME` , then press [Enter]
7. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]
8. At the R> Prompt, type: `SET IDQUOTES=`` , then press [Enter] (the ` quote character is the reverse single quote, which is next to the "1" and under the tilde "~", on American keyboards
9. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
10. Perform the "Complete Database Rebuild" instructions below
11. Repeat "Steps 6-10" for all databases

If there is a Username and Password

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `SET USER UserName Password` , then press [Enter]
7. At the R> Prompt type: `NEWDB DBNAME` , then press [Enter]
8. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]
9. At the R> Prompt, type: `SET IDQUOTES=`` , then press [Enter] (the ` quote character is the reverse single quote, which is next to the "1" and under the tilde "~", on American keyboards
10. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
11. Perform the "Complete Database Rebuild" instructions below
12. Repeat "Steps 6-11" for all databases

If there is a database OWNER

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `SET USER OwnerName` , then press [Enter]
7. At the R> Prompt type: `NEWDB DBNAME` , then press [Enter]
8. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]
9. At the R> Prompt, type: `SET IDQUOTES=`` , then press [Enter] (the ` quote character is the reverse single quote, which is next to the "1" and under the tilde "~", on American keyboards
10. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
11. Perform the "Complete Database Rebuild" instructions below

Copyright © 1982-2024 R:BASE Technologies, Inc.

12. Repeat "Steps 6-11" for all databases

Database Rebuild
In addition to the above conversion steps, the following steps are advised to completely rebuild the database. These steps will be performed in R:BASE 11.

1. Connect to the database and open the R> Prompt.

2. Check the IDQUOTE settings of connected database.

At the R> Prompt, type:

```
SHOW CHARACTERS
```

Notice the last parameter, **IDQUOTES**

If IDQUOTES is set to NULL, you'll need to change NULL to the appropriate character, such as ` (reversed quote) as following:

```
SET IDQUOTES= `
```

3. Unload the entire database. Use the SET USER command below if an OWNER name exists.

At the R> Prompt, type:

```
SET NULL -0-
SET USER ownername
OUTPUT filename.ALL
UNLOAD ALL
OUTPUT SCREEN
```

This process will create two files for Windows databases (filename.ALL and filename.LOB), and a single file for DOS databases.

4. Disconnect the database.

At the R> Prompt, type:

```
DISCONNECT
```

5. Now rename the database that was just unloaded in Steps 1-3.

Review the list of Databases in the R:BASE Database Explorer, and select the "Rename" option. Enter a new name for the database.

6. Finally, use R:BASE to rebuild the database by running the .ALL file.

At the R> Prompt type:

```
RUN filename.ALL
```

These steps will build a clean database!

If there is a considerable amount of errors with both rebuilding the structure and/or data, please refer to the "Database Repair Routine" within the Database Maintenance PDF document in the R:BASE program directory.

Copyright © 1982-2024 R:BASE Technologies, Inc.

With the database file conversion complete, proceed to the chapter for Converting Forms, Reports, & Labels.

1.9.2.2.3 Converting from R:BASE 4.5 through 6.5++

At this point, you must have R:BASE 11 installed on your computer. Be sure to review the R:BASE 11 Getting Started Guide regarding the installation and set up for your R:BASE software. The Getting Started Guide is located in the R:BASE program directory.

If there is no OWNER or Password

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt type: `CONVERT DBNAME` , then press [Enter]
7. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]
8. At the R> Prompt, type: `SET IDQUOTES=` ` , then press [Enter] (the ` quote character is the reverse single quote, which is next to the "1" and under the tilde "~", on American keyboards
9. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
10. Perform the "Complete Database Rebuild" instructions below
11. Repeat "Steps 6-10" for all databases

If there is a Username and Password

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `SET USER UserName Password` , then press [Enter]
7. At the R> Prompt type: `CONVERT DBNAME` , then press [Enter]
8. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]
9. At the R> Prompt, type: `SET IDQUOTES=` ` , then press [Enter] (the ` quote character is the reverse single quote, which is next to the "1" and under the tilde "~", on American keyboards
10. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
11. Perform the "Complete Database Rebuild" instructions below
12. Repeat "Steps 6-11" for all databases

If there is a database OWNER

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `SET USER OwnerName` , then press [Enter]
7. At the R> Prompt type: `CONVERT DBNAME` , then press [Enter]
8. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]
9. At the R> Prompt, type: `SET IDQUOTES=` ` , then press [Enter] (the ` quote character is the reverse single quote, which is next to the "1" and under the tilde "~", on American keyboards
10. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
11. Perform the "Complete Database Rebuild" instructions below
12. Repeat "Steps 6-11" for all databases

Copyright © 1982-2024 R:BASE Technologies, Inc.

Database Rebuild
In addition to the above conversion steps, the following steps are advised to completely rebuild the database. These steps will be performed in R:BASE 11.

1. Connect to the database and open the R> Prompt.

2. Check the IDQUOTE settings of connected database.

At the R> Prompt, type:

```
SHOW CHARACTERS
```

Notice the last parameter, **IDQUOTES**

If IDQUOTES is set to NULL, you'll need to change NULL to the appropriate character, such as ` (reversed quote) as following:

```
SET IDQUOTES= `
```

3. Unload the entire database. Use the SET USER command below if an OWNER name exists.

At the R> Prompt, type:

```
SET NULL -0-
SET USER ownername
OUTPUT filename.ALL
UNLOAD ALL
OUTPUT SCREEN
```

This process will create two files for Windows databases (filename.ALL and filename.LOB), and a single file for DOS databases.

4. Disconnect the database.

At the R> Prompt, type:

```
DISCONNECT
```

5. Now rename the database that was just unloaded in Steps 1-3.

Review the list of Databases in the R:BASE Database Explorer, and select the "Rename" option. Enter a new name for the database.

6. Finally, use R:BASE to rebuild the database by running the .ALL file.

At the R> Prompt type:

```
RUN filename.ALL
```

These steps will build a clean database!

If there is a considerable amount of errors with both rebuilding the structure and/or data, please refer to the "Database Repair Routine" within the Database Maintenance PDF document in the R:BASE program directory.

Copyright © 1982-2024 R:BASE Technologies, Inc.

With the database file conversion complete, proceed to the chapter for Converting Forms, Reports, & Labels.

1.9.2.2.4 Converting from R:BASE 7.x/9.x (32)/10.x

Before converting your R:BASE 7.x/9.x (32)/10.x database(s) with R:BASE 11, the next step is advised if regular database maintenance (PACK/RELOAD) is not performed, to ensure a proper conversion.

The following will create a copy of the database that is completely rebuilt with the structure, data, and indexes reloaded.

1. Launch R:BASE version of 7.x/9.x (32)/10.x
2. Navigate to your database directory by selecting "Utilities" > "Set Current Working Directory..." from the main Menu bar.
3. Go to the R> Prompt window (third button from the left on the Tool Bar)
4. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
5. At the R> Prompt, type: `RENAME DbName.RB* BckupDB.RB*` , then press [Enter]. "DbName" would be the actual name of your R:BASE database.
6. At the R> Prompt, type: `CONNECT BckupDB` , then press [Enter]
7. At the R> Prompt, type: `RELOAD DbName` , then press [Enter]

   The database will now be rebuilt. If database errors occur at any point during the rebuild process, check the integrity of the database files.

8. At the R> Prompt, type: `CONNECT DbName` , then press [Enter]
9. Now connect back to your database after the rebuild, check your database to ensure the contents are intact.
10. After the database is checked, close R:BASE and continue with the steps below.

At this point, you must have R:BASE 11 installed on your computer. Be sure to review the R:BASE 11 Getting Started Guide regarding the installation and set up for your R:BASE software. The Getting Started Guide is located in the R:BASE program directory.

If there is no OWNER

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `CONVERT DBNAME` , then press [Enter]
7. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]

If there is an Owner

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `CONVERT DBNAME IDENTIFIED BY ownername` , then press [Enter]
7. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]

Complete Database Rebuild
In addition to the above conversion steps, the following steps are required to completely rebuild the database. These steps will be performed in R:BASE 11.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1. Connect to the database and open the R> Prompt.

2. Check the IDQUOTE settings of connected database.

At the R> Prompt, type:

```
SHOW CHARACTERS
```

Notice the last parameter, **IDQUOTES**

If IDQUOTES is set to NULL, you'll need to change NULL to the appropriate character, such as ` (reversed quote) as following:

```
SET IDQUOTES= `
```

3. Unload the entire database. Use the SET USER command below if an OWNER name exists.

At the R> Prompt, type:

```
SET NULL -0-
SET USER ownername
OUTPUT filename.ALL
UNLOAD ALL
OUTPUT SCREEN
```

This process will create two files (filename.ALL and filename.LOB)

4. Disconnect the database.

At the R> Prompt, type:

```
DISCONNECT
```

5. Now rename the database that was just unloaded in Steps 1-3.

Review the list of Databases in the R:BASE Database Explorer, and select the "Rename" option. Enter a new name for the database.

6. Finally, use R:BASE to rebuild the database by running the .ALL file.

At the R> Prompt type:

```
RUN filename.ALL
```

These steps will build a clean database!

With the database file conversion complete, proceed to the chapter for converting forms, reports, and labels. Choose either of the below:

R:BASE 7.x/Turbo V-8 Forms, Reports, & Labels
R:BASE eXtreme 9.x (32/64) Forms, Reports, & Labels

1.9.2.2.5 Converting from R:BASE Turbo V-8

Before attempting to convert your R:BASE Turbo V-8 database(s) with R:BASE 11, the next step is advised if regular database maintenance (PACK/RELOAD) is not performed, to ensure a proper conversion.

Copyright © 1982-2024 R:BASE Technologies, Inc.

The following will create a copy of the database that is completely rebuilt with the structure, data, and indexes reloaded.

1. Launch R:BASE Turbo V-8
2. Navigate to your database directory by selecting "Utilities" > "Set Current Working Directory..." from the main Menu bar.
3. Go to the R> Prompt window (third button from the left on the Tool Bar)
4. At the R> Prompt, type: `DISCONNECT` , then press [Enter]
5. At the R> Prompt, type: `RENAME DbName.RX* BckupDB.RX*` , then press [Enter]. "DbName" would be the actual name of your R:BASE database.
6. At the R> Prompt, type: `CONNECT BckupDB` , then press [Enter]
7. At the R> Prompt, type: `RELOAD DbName` , then press [Enter]

   The database will now be rebuilt. If database errors occur at any point during the rebuild process, check the integrity of the database files.

8. At the R> Prompt, type: `CONNECT DbName` , then press [Enter]
9. Now connect back to your database after the rebuild, check your database to ensure the contents are intact.
10. After the database is checked, close R:BASE and continue with the steps below.

At this point, you must have R:BASE 11 installed on your computer. Be sure to review the R:BASE 11 Getting Started Guide regarding the installation and set up for your R:BASE software. The Getting Started Guide is located in the R:BASE program directory.

If there is no OWNER

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `TURBO DBNAME V9` , then press [Enter]
7. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]

If there is an Owner

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `TURBO DBNAME V9 IDENTIFIED BY ownername` , then press [Enter]
7. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]

Complete Database Rebuild
In addition to the above conversion steps, the following steps are required to completely rebuild the database. These steps will be performed in R:BASE 11.

1. Connect to the database and open the R> Prompt.

2. Check the IDQUOTE settings of connected database.

At the R> Prompt, type:

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
SHOW CHARACTERS
```

Notice the last parameter, **IDQUOTES**

If IDQUOTES is set to NULL, you'll need to change NULL to the appropriate character, such as ` (reversed quote) as following:

```
SET IDQUOTES= `
```

3. Unload the entire database. Use the SET USER command below if an OWNER name exists.

At the R> Prompt, type:

```
SET NULL -0-
SET USER ownername
OUTPUT filename.ALL
UNLOAD ALL
OUTPUT SCREEN
```

This process will create two files (filename.ALL and filename.LOB)

4. Disconnect the database.

At the R> Prompt, type:

```
DISCONNECT
```

5. Now rename the database that was just unloaded in Steps 1-3.

Review the list of Databases in the R:BASE Database Explorer, and select the "Rename" option. Enter a new name for the database.

6. Finally, use R:BASE to rebuild the database by running the .ALL file.

At the R> Prompt type:

```
RUN filename.ALL
```

These steps will build a clean database!

With the database file conversion complete, proceed to the chapter for converting R:BASE 7.x/Turbo V-8 Forms, Reports, & Labels.

1.9.2.2.6 Converting from R:BASE eXtreme 9.x (64)

Before attempting to convert your R:BASE eXtreme 9.x (64) database(s) with R:BASE 11, the next step is advised if regular database maintenance (PACK/RELOAD) is not performed, to ensure a proper conversion.

The following will create a copy of the database that is completely rebuilt with the structure, data, and indexes reloaded.

1. Launch R:BASE eXtreme 9.5 (64)
2. Navigate to your database directory by selecting "Utilities" > "Set Current Working Directory..." from the main Menu bar.
3. Go to the R> Prompt window (third button from the left on the Tool Bar)
4. At the R> Prompt, type: `DISCONNECT` , then press [Enter]

Copyright © 1982-2024 R:BASE Technologies, Inc.

5. At the R> Prompt, type: `RENAME DbName.RX* BckupDB.RX*` , then press [Enter]. "DbName" would be the actual name of your R:BASE database.
6. At the R> Prompt, type: `CONNECT BckupDB` , then press [Enter]
7. At the R> Prompt, type: `RELOAD DbName` , then press [Enter]

   The database will now be rebuilt. If database errors occur at any point during the rebuild process, check the integrity of the database files.

8. At the R> Prompt, type: `CONNECT DbName` , then press [Enter]
9. Now connect back to your database after the rebuild, check your database to ensure the contents are intact.
10. After the database is checked, close R:BASE and continue with the steps below.

At this point, you must have R:BASE 11 installed on your computer. Be sure to review the R:BASE 11 Getting Started Guide regarding the installation and set up for your R:BASE software. The Getting Started Guide is located in the R:BASE program directory.

If there is no OWNER

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `MIGRATE DBNAME` , then press [Enter]
7. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]

If there is an Owner

1. Install and launch R:BASE 11
2. Navigate to your database directory using the "Change Current Folder" button (bottom left corner, folder icon with a green arrow, at the Database Explorer)
3. Go to the R> Prompt window (third button from the left on the Tool Bar, or press [Ctrl]+[R])
4. At the R> Prompt, type: `SET MULTI OFF` , then press [Enter]
5. At the R> Prompt, type: `SET STATICDB OFF` , then press [Enter]
6. At the R> Prompt, type: `SET USER OwnerName` , then press [Enter]
7. At the R> Prompt, type: `MIGRATE DBNAME` , then press [Enter]
8. At the R> Prompt, type: `CONNECT DBNAME` , then press [Enter]

Complete Database Rebuild
In addition to the above conversion steps, the following steps are required to completely rebuild the database. These steps will be performed in R:BASE 11.

1. Connect to the database and open the R> Prompt.

2. Check the IDQUOTE settings of connected database.

At the R> Prompt, type:

```
SHOW CHARACTERS
```

Notice the last parameter, **IDQUOTES**

If IDQUOTES is set to NULL, you'll need to change NULL to the appropriate character, such as ` (reversed quote) as following:

```
SET IDQUOTES= `
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

3. Unload the entire database. Use the SET USER command below if an OWNER name exists.

At the R> Prompt, type:

```
SET NULL -0-
SET USER ownername
OUTPUT filename.ALL
UNLOAD ALL
OUTPUT SCREEN
```

This process will create two files (filename.ALL and filename.LOB)

4. Disconnect the database.

At the R> Prompt, type:

```
DISCONNECT
```

5. Now rename the database that was just unloaded in Steps 1-3.

Review the list of Databases in the R:BASE Database Explorer, and select the "Rename" option. Enter a new name for the database.

6. Finally, use R:BASE to rebuild the database by running the .ALL file.

At the R> Prompt type:

```
RUN filename.ALL
```

These steps will build a clean database!

With the database file conversion complete, proceed to the chapter for converting R:BASE eXtreme 9.x (32/64) Forms, Reports, & Labels.

1.9.2.2.7 Converting from R:BASE X/X.5 Enterprise (10.x)

If you are upgrading to R:BASE 11 from R:BASE X/X.5 Enterprise (Version 10.x), you are only required to connect to the database in order to complete the conversion.

At this point in R:BASE use, R:BASE 11 and R:BASE X/X.5 Enterprise can connect to the same database. However, once any Version 11 specific features are implemented (such as static variables or database comments), R:BASE 10.x Enterprise versions can no longer connect to the database. A minor version number is stored within file 1 of the database files. R:BASE 11 will automatically alter the value if any new features of version 11 are implemented. Also, any new databases created in R:BASE 11 will start with the elevated minor version, and R:BASE 10.x Enterprise versions will not be able to connect to the database.

Please review the "What's New in R:BASE 11 for Windows" PDF document for the list of other new features.

1.9.3 Converting Forms, Reports, & Labels

At this point in the conversion process, four database files should exist. When connected to a database, tables will be listed in the R:BASE Database Explorer window. You should also be able to browse the table data with no problems.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Now, steps can be taken to convert the existing forms, reports, and labels to R:BASE 11 for Windows. Whether the database originates from DOS or Windows all forms, reports, and labels can easily be converted to R:BASE 11.

Proceed to the next chapter for converting 6.5++ and Lower Version Forms, Reports, & Labels.

1.9.3.1 R:BASE 6.5++ and Lower Forms, Reports, & Labels

At this point in the database conversion process, R:BASE 11 for Windows must be installed on the computer to complete the instructions below.

1. Launch R:BASE 11 for Windows
2. Connect to the database by selecting "Database" > "Connect to Database..." from the main Menu Bar.

   You will be prompted to locate your database files where they are stored on your computer. Once connected to your database, you will see the database name listed across the top of the R:BASE 11 program caption, in brackets.

Throughout the following instructions, there will be areas of the R:BASE interface that will require you to know what is being referred to (e.g. Database Explorer, Navigator, Tool Bar, Menu Bar, etc.). For a quick description as to what these objects are, please read the "Introduction" page of the R:BASE 11 Tutorial before continuing. To open the **Introduction** page, select "Help" > "R:BASE Tutorial" from the R:BASE main Menu Bar.

Once you have completed reading the Introduction page, close the window and go to the next section, **Converting Forms**.

1.9.3.1.1 Converting Forms

You should now be looking at the R:BASE window with the Database Explorer displayed. To confirm, you will see the main R:BASE caption read the "R:BASE 11", then your database name listed in square brackets, then "R:BASE Database Explorer" listed in square brackets.

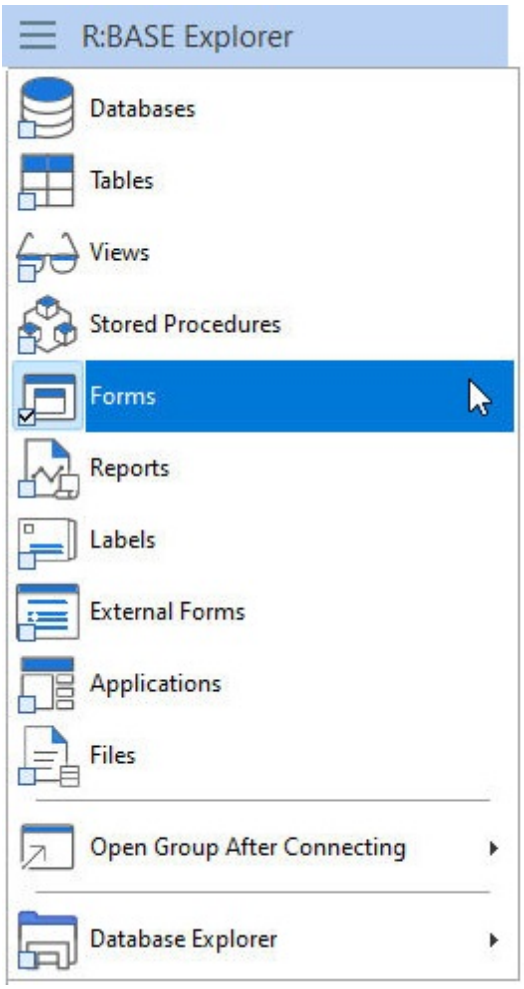
- If you do not see your database listed in brackets, then connect to it now by selecting "Database" > "Connect to Database..." from the main Menu Bar. You will be prompted to locate your database files where they are stored on your computer. Once connected to your database, you will see the database name listed in the R:BASE 11 program caption, in brackets.

- If you do not see "R:BASE Database Explorer" listed in square brackets, then select the "Database Explorer" button on the main Tool Bar. It is the first button on the left and the hint displayed will read "Database Explorer".

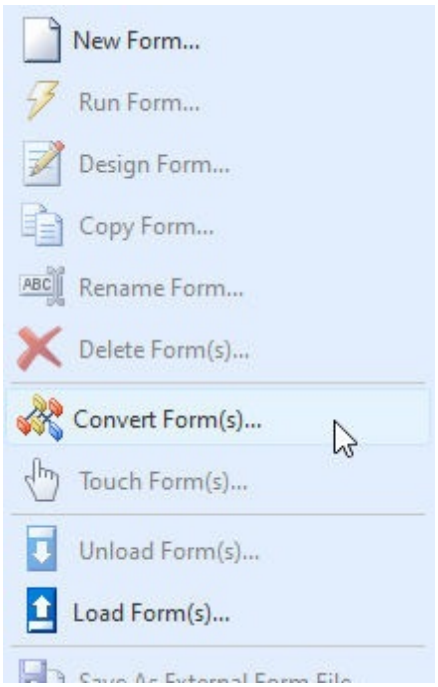
At the Database Explorer window, you will see the "Navigator" listed on the left side of the window containing various menu options of R:BASE database modules (e.g. Tables, Views, Forms, etc.)

1. Click on the Database Explorer button and choose "Forms" from the drop down menu (figure 1), then select the "Convert Form(s)..." menu option (figure 2).

Copyright © 1982-2024 R:BASE Technologies, Inc.



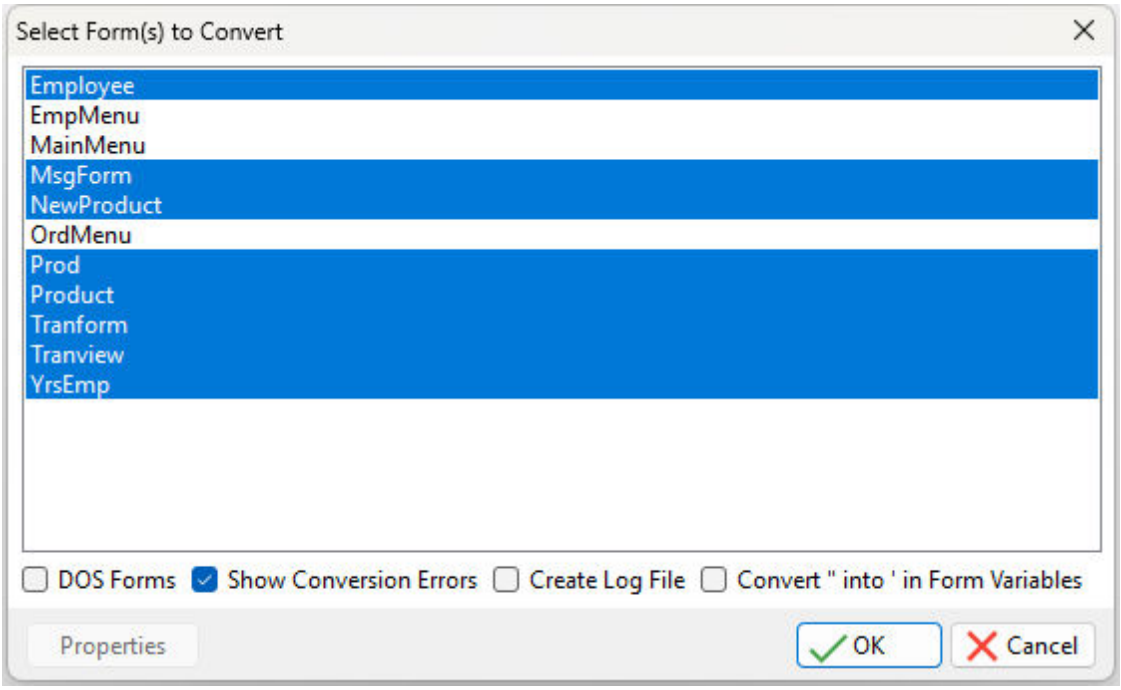
(figure 1)



(figure 2)

Upon selecting this option, a dialog will appear that may contain a list of the available forms to convert (figure 3). The forms listed are Windows-based forms that were created in a previous Windows version of R:BASE.

Copyright © 1982-2024 R:BASE Technologies, Inc.



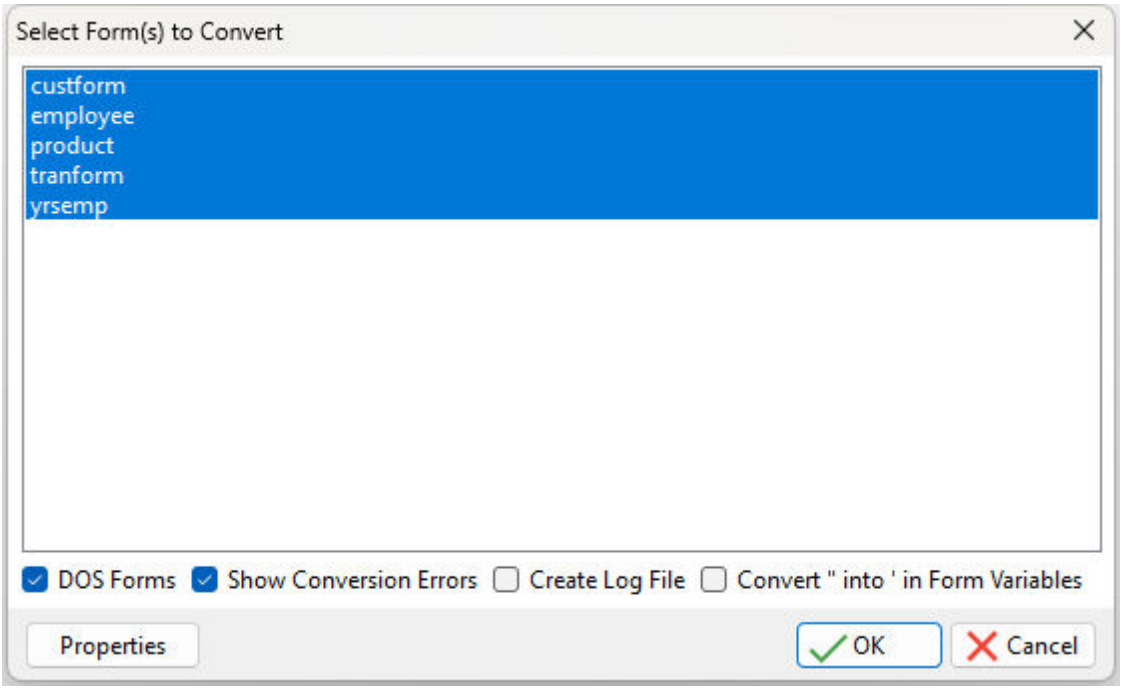
(figure 3)

2. Individual forms can be selected and converted. Multiple forms can be converted by holding the [Ctrl] key and selecting individual form names. Or pressing the [Ctrl]+[A] keys will select all listed forms. Press the "OK" button to convert the selected forms. Conversion options include the ability to show errors, create a log file, and convert " into ' for form variables as check boxes. As the earlier steps advised to change the QUOTES setting from double quotes to an apostrophe, the last option will make legacy migrations much easier.

   The form(s) will convert and will be displayed in the Database Explorer.

3. The database may contain DOS forms. Select the "DOS Forms" check box. The dialog may now contain a list of the available DOS forms to convert (figure 4). Any forms listed are DOS-based forms that were created in a previous DOS version of R:BASE.

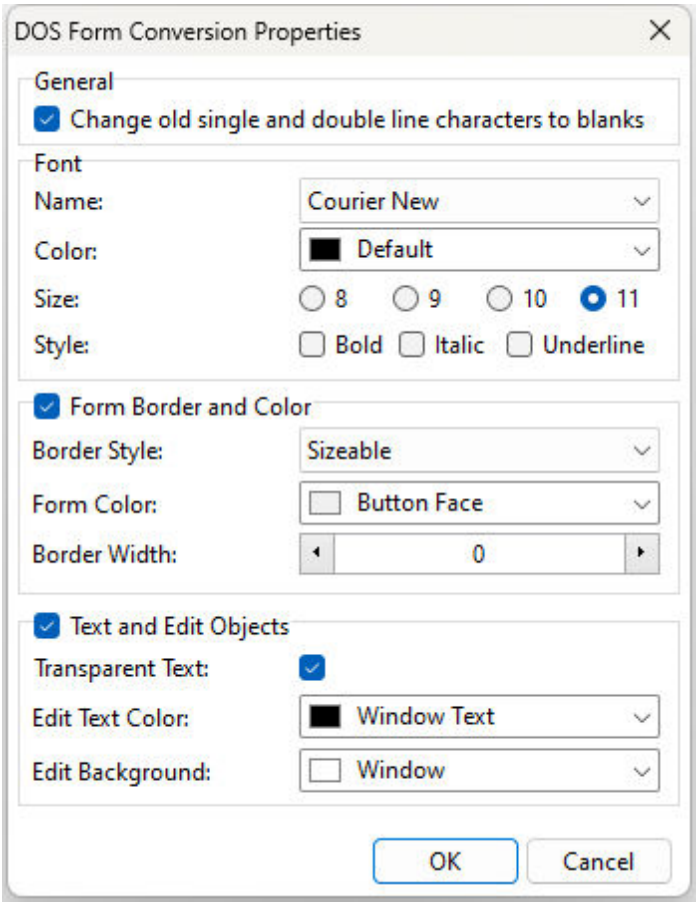
Copyright © 1982-2024 R:BASE Technologies, Inc.



(figure 4)

4. The "Properties" button provides a form conversion option to change single and double line characters to blank spaces, alter the Font (name, color, size, and style), Form Border (style and width), Form Background Color, Text (transparent), and Edit Objects (text color and background) (figure 5).

Copyright © 1982-2024 R:BASE Technologies, Inc.



(figure 5)

The following is an example of a form that was converted with the single and double line characters setting unchecked. This display may be beneficial in the conversion process to see how a form may have been divided into sections.

Copyright © 1982-2024 R:BASE Technologies, Inc.

5. Like with the Windows-based forms, the DOS forms can be individually selected and converted. Multiple forms can be converted by holding the [Ctrl] key and selecting individual form names. Or pressing the [Ctrl]+[A] keys will select all listed forms. Press the "OK" button to convert the selected forms.

At this point, you can now enter the Form Designer to view and make changes to the forms. There are Issues and Suggestions to review for the converted forms. However, since many of the instructions below refer to specific control/object names and require a basic understanding of the Form Designer, it is recommended that you perform the conversions of the remaining modules (e.g. reports, labels), and then complete the R:BASE Tutorial before continuing.

From the Main Menu, choose "Help" > "R:BASE Tutorial". This tutorial covers step-by-step instructions on building a complete database and application from scratch providing the R:BASE relational logic as it adhere's to Dr. Codd's relational model. This is where one would learn how to quickly adapt to the R:BASE interface!

1.9.3.1.1.1 Issues and Suggestions

NOTE: From inside the Form Designer, you can launch the Form Designer help file by pressing the [Shift]+[F1] hot keys.

- **Add an Enhanced DB Navigator Control**

  The Enhanced DB Navigator is a form object that allows moving between, saving, and deleting rows in a table. To add the control, locate the button within the Database Controls (boxed in red below).

Copyright © 1982-2024 R:BASE Technologies, Inc.

Mouse click on the button, then mouse click on the form where the object is to be placed. The properties dialog will appear to alter the settings for the control. Click OK to use the default settings.

- **Form Lookup Variables**
  The "Lookup Variable" is now truly a "Lookup" variable, and is not to be changed. All expressions, if based on lookup variables, will be evaluated and refreshed INSTANTLY. Prior to version 7.x, this was not the case. Previously, the form variables must be refreshed with the RECALC VARIABLES command each time variables were to be refreshed. To achieve the same results in R:BASE 11, then do NOT use form defined variables, but rather EEPs to declare variables with the SET VARIABLE command, which can execute when needed and with the flexibility to change the variables accordingly.

- **Form Variables**
  Form variables that perform calculations or combine a number of values should be enclosed in parenthesis. If error messages are displayed for variables performing calculations, like the one below,

```
2 : CURRENCY    vTotal = price * quantity
```

  add parenthesis as follows:

```
2 : CURRENCY    vTotal = (price * quantity)
```

- **Multi-Page Form with Push Buttons**
  When converting a multiple page form, the different "pages" will be placed on a Tab Control, with each page placed upon a separate "tab". Any Push Buttons on the form that use a predefined "Next Page" or "Previous Page" EEP will appear on the new form, however will have no action assigned to them. This is because there is technically no longer a "page" to move to with regards to the behavior of a Tab Control.

- **Lookup Variable Tip**
  Within the R:BASE 11 Form Designer there is an option to run a series of commands (or Action) for a form before the Form Designer is launched. In the Form Designer, choose "Layout" from the Menu bar, the select "On Before Design Action...".

  This will run a series of commands, which can include the creation of variables that Lookup Variables are based upon, to avoid annoying error messages when the form is opened in the designer. An example of a lookup variable based upon another variable would be...

```
1 : TEXT    vCompany = Company IN Customer WHERE CustID =
.vCustID
```

  The error message appears because the variable vCustID is not defined. Adding the following SET VAR command to the "On Before Design Action..." will alleviate this error message:

```
SET VAR vCustID INTEGER
```

  The "On Before Design Action..." is listed under "Layout" on the main Menu Bar.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.9.3.1.2 Converting Reports

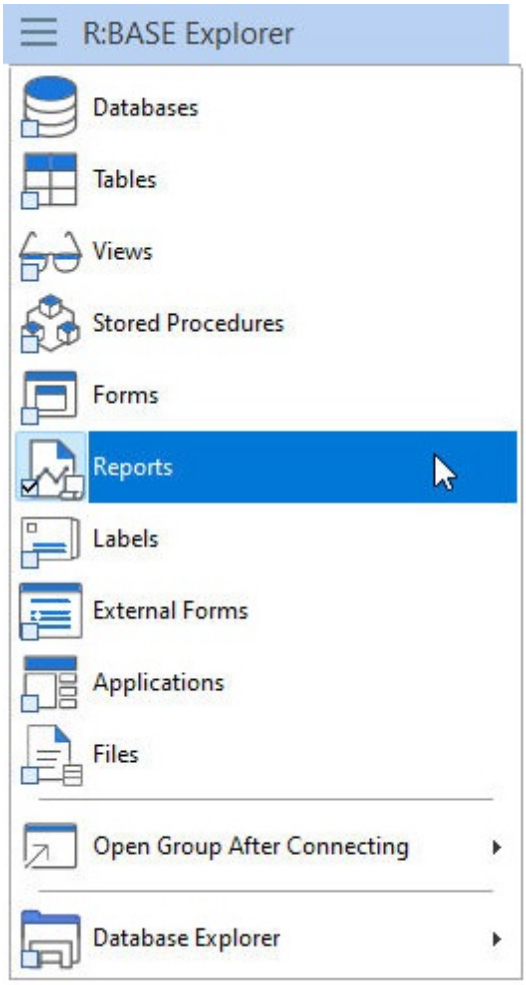
You should now be looking at the R:BASE window with the Database Explorer displayed. To confirm, you will see the main R:BASE caption read the "R:BASE 11", then your database name listed in square brackets, then "R:BASE Database Explorer" listed in square brackets.

- If you do not see your database listed in brackets, then connect to it now by selecting "Database" > "Connect to Database..." from the main Menu Bar. You will be prompted to locate your database files where they are stored on your computer. Once connected to your database, you will see the database name listed in the R:BASE 11 program caption, in brackets.

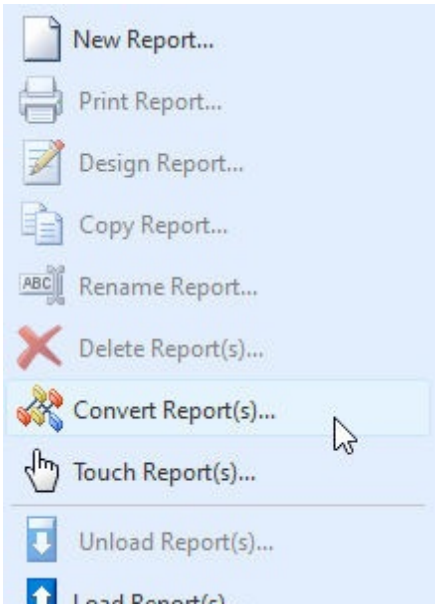
- If you do not see "R:BASE Database Explorer" listed in square brackets, then select the "Database Explorer" button on the main Tool Bar. It is the first button on the left and the hint displayed will read "Database Explorer".

At the Database Explorer window, you will see the "Navigator" listed on the left side of the window containing various menu options of R:BASE database modules (e.g. Tables, Views, Forms, etc.)

1. Click on the Database Explorer button and choose "Reports" from the drop down menu (figure 1), then select the "Convert Report(s)..." menu option (figure 2).



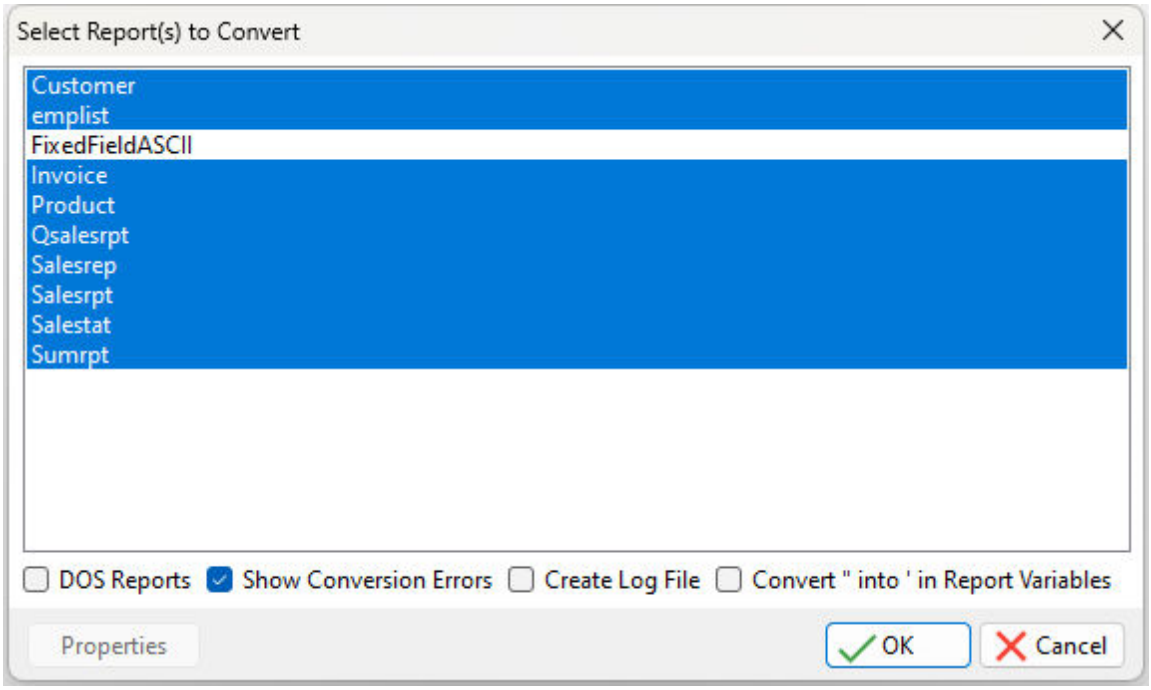
(figure 1)



(figure 2)

Copyright © 1982-2024 R:BASE Technologies, Inc.

Upon selecting this option, a dialog will appear that may contain a list of the available reports to convert (figure 3). The reports listed are Windows-based reports that were created in a previous Windows version of R:BASE.



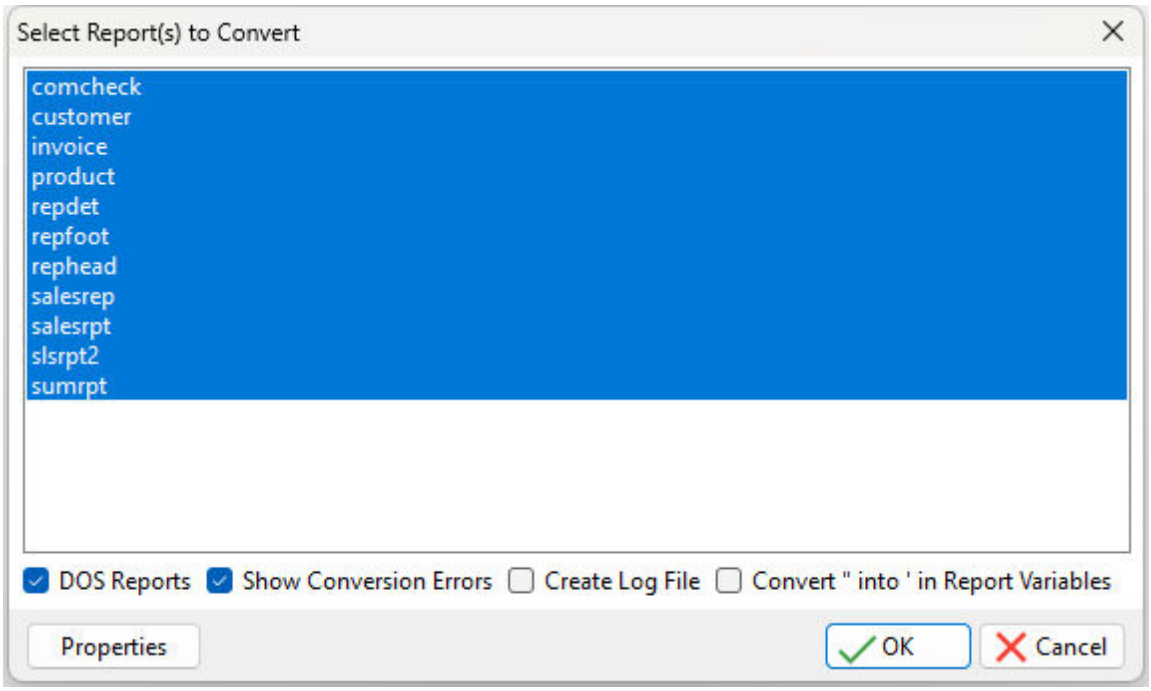
(figure 3)

2. Individual reports can be selected and converted. Multiple reports can be converted by holding the [Ctrl] key and selecting individual report names. Or pressing the [Ctrl]+[A] keys will select all listed reports. Press the "OK" button to convert the selected reports. Conversion options include the ability to show errors, create a log file, and convert " into ' for report variables as check boxes. As the earlier steps advised to change the QUOTES setting from double quotes to an apostrophe, the last option will make legacy migrations much easier.

   The report(s) will convert and will be displayed in the Database Explorer.

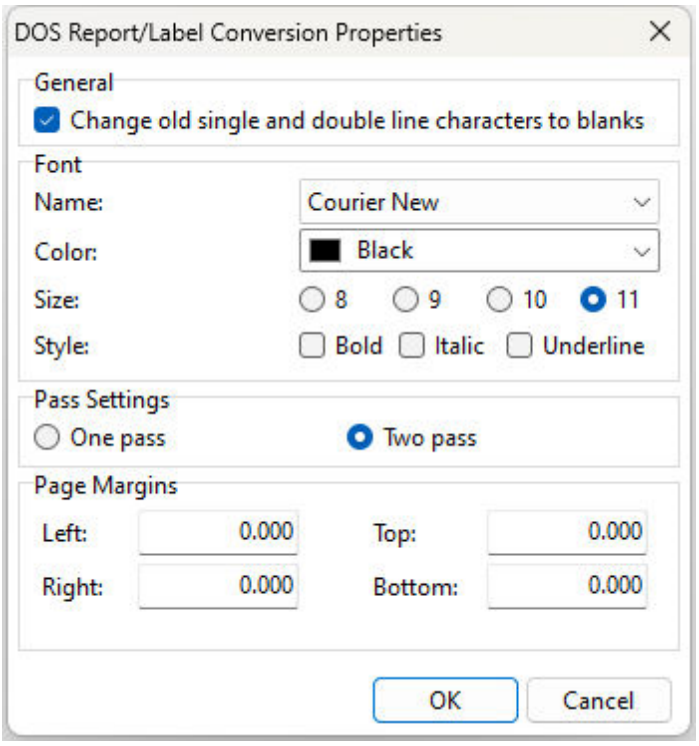
3. The database may contain DOS reports. Select the "DOS Reports" check box. The dialog may now contain a list of the available DOS reports to convert (figure 4). Any reports listed are DOS-based reports that were created in a DOS version of R:BASE.

Copyright © 1982-2024 R:BASE Technologies, Inc.



(figure 4)

4. The "Properties" button provides a form conversion option to change single and double line characters to blank spaces, alter the Font (name, color, size, and style), Form Border (style and width), Form Background Color, Text (transparent), and Edit Objects (text color and background) (figure 5).



(figure 5)

Copyright © 1982-2024 R:BASE Technologies, Inc.

Maintaining single and double lines in the conversion process may help see how a report was divided into sections. The pass setting processes system variables once or twice are the report is generated. For DOS reports, the default is one pass. For Windows reports, the default pass setting is two. Margins in DOS reports begin at zero. The margin can be changed for any custom reports.

5. Like with the Windows-based reports, the DOS reports can be individually selected and converted. Multiple reports can be converted by holding the [Ctrl] key and selecting individual report names. Or pressing the [Ctrl]+[A] keys will select all listed reports. Press the "OK" button to convert the selected reports.

   The report(s) will convert and will be displayed within the Database Explorer.

At this point, you can now enter the Report Designer to view and make changes to the reports. There are Issues and Suggestions to review for the converted reports. However, since many of the instructions below refer to specific control/object names and require a basic understanding of the Report Designer, it is recommended that you perform the conversions of the remaining modules (e.g. forms, labels), and then complete the R:BASE Tutorial before continuing.

From the Main Menu, choose "Help" > "R:BASE Tutorial". This tutorial covers step-by-step instructions on building a complete database and application from scratch providing the R:BASE relational logic as it adhere's to Dr. Codd's relational model. This is where one would learn how to quickly adapt to the R:BASE interface!

1.9.3.1.2.1 Issues and Suggestions

NOTE: From inside the Report Designer, you can launch the Report Designer help file by pressing the [Shift]+[F1] hot keys.

- **Sub-Totals and Totals**
  To calculate sub-totals and totals of table columns, Aggregate Variables have always been used to perform this job. Now, the R:BASE 11 for Windows Report Designer has a built-in object called DB Calc, which performs these same functions on table columns without creating the extra overhead of variables. You can replace your Variable Label objects with DB Calc objects in instances where the Variables Object results will not be used further down in the report.

- **#DATE, #TIME, #PAGE, Document, and Page Information**
  The R:BASE 11 for Windows Report Designer now has a System Variable control, which allows an object to be placed for the current DATE, TIME, Document Name, Print DATE and TIME, or one of many Page Numbering options. All Variable Label objects whose expression value consists of a system variable (#DATE, #TIME) should be deleted and replaced with a System Variable control. Any Variable Label objects whose expression value consists of the #PAGE system variable, you must delete the variable object and replace that object with a System Variable control.

- **Report Variables**
  Report variables that perform calculations or combine a number of values should be enclosed in parenthesis. If error messages are displayed for variables performing calculations, like the one below,

  ```
  2 : CURRENCY    vTotal = price * quantity
  ```

  add parenthesis as follows:

  ```
  2 : CURRENCY    vTotal = (price * quantity)
  ```

- **Report Totals**
  To be sure report totals are calculating correctly on all NULL and not NULL records, make sure the **ZERO** setting is ON. Check the setting under "Settings" > "Configuration Settings".

Copyright © 1982-2024 R:BASE Technologies, Inc.

- **Printer Control Codes**
  If DOS or machine printer control codes are used in R:BASE Report variables, they can no longer be stored in the Report Variables. A control code would appear in the variable list with an expression as follows: <27 40 115 49 83>

  If the control code were used to alter the text on the report (e.g. bold, underline, etc.), the control code must be deleted. There are many new features in the Windows interface that replace control codes for altering report text objects. The R:BASE 11 for Windows version allows you to perform these font changes must easier.

  If control codes listed in the report control hardware for paper feeds or to open a cash drawer, then document down the exact code and review the "PCC Label" Control in the Report Designer. This control will support the same functionality used previously.

- **Picture Format**
  For a report object, **[]** may be displayed in the report preview. In R:BASE for DOS and R:BASE for Windows versions prior to 7.x, a "Picture Format" option was available. The option allowed the justification of the value and several formatting options based upon the data type of the object. For example, to add the justification a [>], or [<] was used to right/left justify.

  In R:BASE 11, this feature is called "Display Format", which still offers several formatting options for report objects based upon the data type of the object. While the formatting option is still supported, the justification option is no longer supported in the previous context, within the "Display Format" settings. To right/left/center justify text of an object, use the "Format" tool bar which supports many text formatting options.

  If you see **[]** listed in your report preview for a report object, you must right click on the object, select "Display Format", and remove the [] characters.

- **Page Setup**
  For DOS to Windows conversions, the default page size setting is set to it's largest values in order to make sure no objects are lost in the conversion process. You must make sure you alter your page settings back to your desired values in order for your reports to print correctly.

  To alter the page size settings, follow the below instructions:

  1. Select "File" > "Page Setup..." from the main Menu Bar.
  2. Select the "Paper Size" tab
  3. For Portrait orientation reports, the Width should be set to "8.5".
  4. For Landscape orientation reports, the Height should be set to "8.5".

     From this dialog, you can also adjust your report margins from the "Margins" tab.

  After making any changes, make sure that all of your report objects are located within the defined page size settings and margins. Otherwise, R:BASE will display a warning that "Controls beyond the new page width will be deleted." In this case, select "Cancel" to avoid losing any controls. Then move your controls within your desired page settings and then make your page size setting changes again.

- **Report Preview Tip**
  The "Report Preview" tab will display a preview of your report as it may appear when you send the report to the printer or screen. Before selecting the tab, you must make sure that there are no errors listed in your Report variables, otherwise, you will receive an error message for every error listed multiplied by the number of records displayed in the Report Preview window.

  What you can do to limit the number of rows displayed in the Report Preview window, thus limiting the number error messages. To do so, alter the Report Default settings for how many rows are displayed when previewing a report.

  1. Select "Settings" > "Report/Label Designer" > "Default Settings..." from the main Menu Bar.
  2. Alter the value for "Row Count for Preview:" to a smaller value other than 100; perhaps to 10.

Copyright © 1982-2024 R:BASE Technologies, Inc.

- **Lookup Variable Tip**
  Within the R:BASE 11 Report Designer there is an option to run a series of commands (or Action) for a report before the Report Designer is launched. In the Report Designer, choose "Report" from the Menu bar, the select "Actions" > "On Before Design...".

  This will run a series of commands, which can include the creation of variables that Lookup Variables are based upon, to avoid annoying error messages when the form is opened in the designer. An example of a lookup variable based upon another variable would be...

```
1 : TEXT    vCompany = Company IN Customer WHERE CustID =
.vCustID
```

  The error message appears because the variable vCustID is not defined. Adding the following SET VAR command to the "On Before Design Action..." will alleviate this error message:

```
SET VAR vCustID INTEGER
```

  The "On Before Design..." action is located under "Report" > "Actions" on the main Menu Bar.

1.9.3.1.3 Converting Labels

You should now be looking at the R:BASE window with the Database Explorer displayed. To confirm, you will see the main R:BASE caption read the "R:BASE 11", then your database name listed in square brackets, then "R:BASE Database Explorer" listed in square brackets.

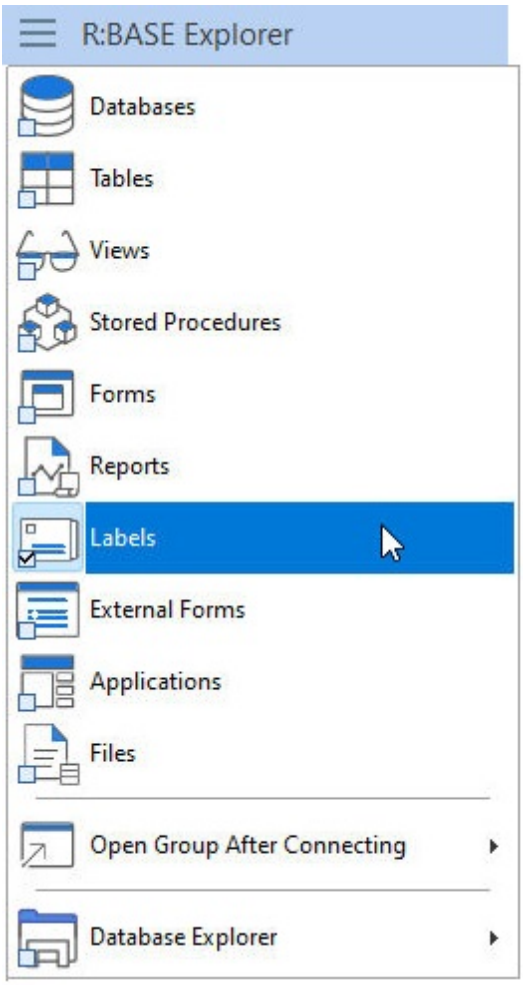
- If you do not see your database listed in brackets, then connect to it now by selecting "Database" > "Connect to Database..." from the main Menu Bar. You will be prompted to locate your database files where they are stored on your computer. Once connected to your database, you will see the database name listed in the R:BASE 11 program caption, in brackets.

- If you do not see "R:BASE Database Explorer" listed in square brackets, then select the "Database Explorer" button on the main Tool Bar. It is the first button on the left and the hint displayed will read "Database Explorer".

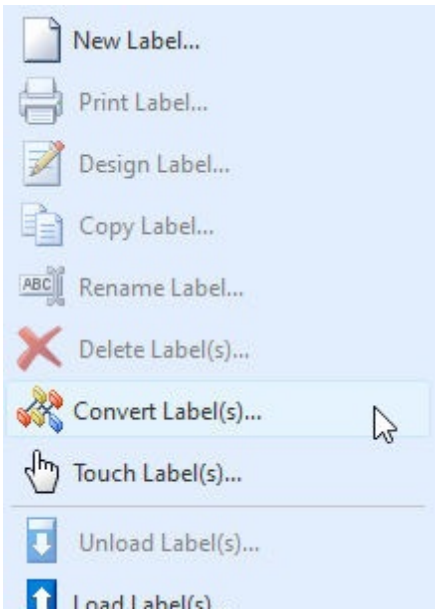
At the Database Explorer window, you will see the "Navigator" listed on the left side of the window containing various menu options of R:BASE database modules (e.g. Tables, Views, Forms, etc.)

1. Click on the Database Explorer button and choose "Labels" (figure 1), then select the "Convert Label(s)..." sub menu option (figure 2).

Copyright © 1982-2024 R:BASE Technologies, Inc.



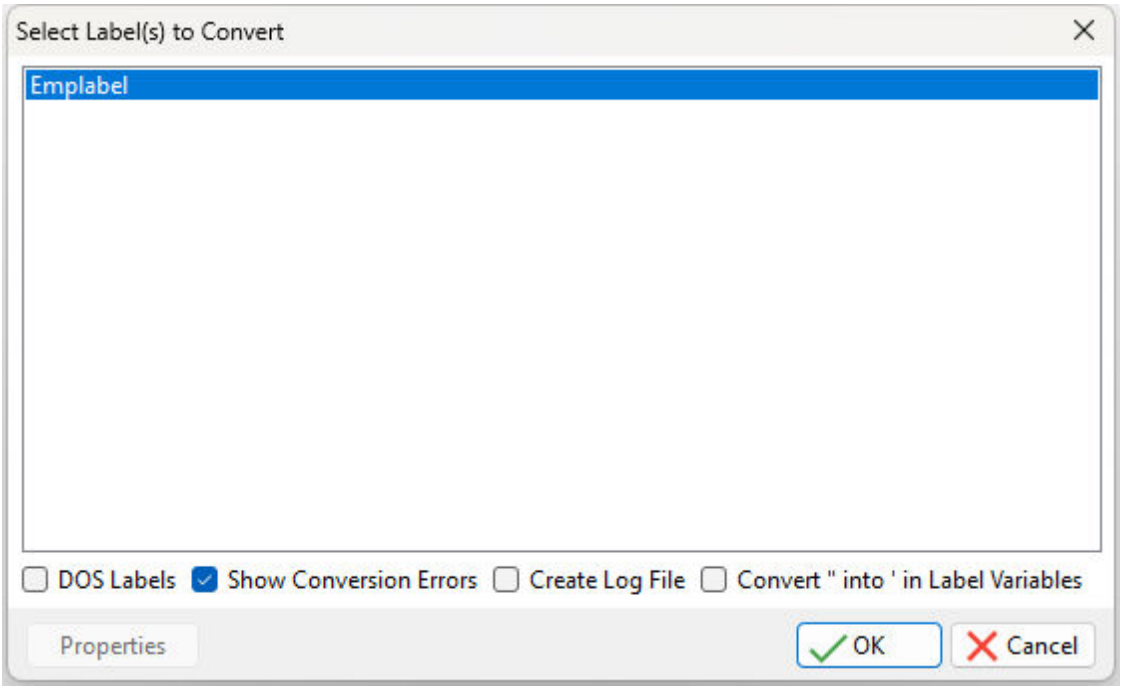
(figure 1)



(figure 2)

Upon selecting this option, a dialog will appear that may contain a list of the available labels to convert (figure 3). The labels listed are Windows-based labels that were created in a previous Windows version of R:BASE.

Copyright © 1982-2024 R:BASE Technologies, Inc.



(figure 3)

2. Individual labels can be selected and converted. Multiple labels can be converted by holding the [Ctrl] key and selecting individual label names. Or pressing the [Ctrl]+[A] keys will select all listed labels. Press the "OK" button to convert the selected labels. Conversion options include the ability to show errors, create a log file, and convert " into ' for label variables as check boxes. As the earlier steps advised to change the QUOTES setting from double quotes to an apostrophe, the last option will make legacy migrations much easier.

   The label(s) will convert and will be displayed in the Database Explorer.

3. The database may contain DOS labels. Select the "DOS Labels" check box. The dialog may now contain a list of the available DOS labels to convert. Any labels listed are DOS-based labels that were created in a DOS version of R:BASE. The "Properties" button provides label conversion options to change single and double line characters to blanks, a pass setting, and margins (figure 5).

   Maintaining single and double lines in the conversion process may help see how a label was divided into sections. The pass setting processes system variables once or twice are the label is generated. For DOS labels, the default is one pass. For Windows labels, the default pass setting is two. Margins in DOS labels begin at zero. The margin can be changed for any custom reports.

4. Like with the Windows-based labels, the DOS labels can be individually selected and converted. Multiple labels can be converted by holding the [Ctrl] key and selecting individual label names. Or pressing the [Ctrl]+[A] keys will select all listed labels. Press the "OK" button to convert the selected labels.

   The label(s) will convert and will be displayed within the Database Explorer.

At this point, you can now enter the Label Designer to view and make changes to the labels. There are Issues and Suggestions to review for the converted labels. However, since many of the instructions below refer to specific control/object names and require a basic understanding of the Label Designer, it is recommended that you perform the conversions of the remaining modules (e.g. forms, reports), and then complete the R:BASE Tutorial before continuing.

From the Main Menu, choose "Help" > "R:BASE Tutorial". This tutorial covers step-by-step instructions on building a complete database and application from scratch providing the R:BASE relational logic as it

Copyright © 1982-2024 R:BASE Technologies, Inc.

adhere's to Dr. Codd's relational model. This is where one would learn how to quickly adapt to the R:BASE interface!

1.9.3.1.3.1 Issues and Suggestions

NOTE: From inside the Label Designer, you can launch the Label Designer help file by pressing the [Shift] +[F1] hot keys.

- **Sub-Totals and Totals**
  To calculate sub-totals and totals of table columns, Aggregate Variables have always been used to perform this job. Now, the R:BASE 11 for Windows Label Designer has a built-in object called DB Calc, which performs these same functions on table columns without creating the extra overhead of variables. You can replace your Variable Label objects with DB Calc objects in instances where the Variables Object results will not be used further down in the label.

- **#DATE, #TIME, #PAGE, Document, and Page Information**
  The R:BASE 11 for Windows Label Designer now has a System Variable control, which allows an object to be placed for the current DATE, TIME, Document Name, Print DATE and TIME, or one of many Page Numbering options. All Variable Label objects whose expression value consists of a system variable (#DATE, #TIME) should be deleted and replaced with a System Variable control. Any Variable Label objects whose expression value consists of the #PAGE system variable, you must delete the variable object and replace that object with a System Variable control.

- **Label Variables**
  Label variables that perform calculations or combine a number of values should be enclosed in parenthesis. If error messages are displayed for variables performing calculations, like the one below,

  ```
  2 : CURRENCY    vTotal = price * quantity
  ```

  add parenthesis as follows:

  ```
  2 : CURRENCY    vTotal = (price * quantity)
  ```

- **Label Totals**
  To be sure label totals are calculating correctly on all NULL and not NULL records, make sure the **ZERO** setting is ON. Check the setting under "Settings" > "Configuration Settings".

- **Printer Control Codes**
  If DOS or machine printer control codes are used in R:BASE Label variables, they can no longer be stored in the Label Variables. A control code would appear in the variable list with an expression as follows: <27 40 115 49 83>

  If the control code were used to alter the text on the label (e.g. bold, underline, etc.), the control code must be deleted. There are many new features in the Windows interface that replace control codes for altering label text objects. The R:BASE 11 for Windows version allows you to perform these font changes must easier.

  If control codes listed in the label control hardware for paper feeds or to open a cash drawer, then document down the exact code and review the "PCC Label" Control in the Label Designer. This control will support the same functionality used previously.

- **Picture Format**
  For a label object, **[]** may be displayed in the label preview. In R:BASE for DOS and R:BASE for Windows versions prior to 7.x, a "Picture Format" option was available. The option allowed the justification of the value and several formatting options based upon the data type of the object. For example, to add the justification a [>], or [<] was used to right/left justify.

Copyright © 1982-2024 R:BASE Technologies, Inc.

In R:BASE 11, this feature is called "Display Format", which still offers several formatting options for report objects based upon the data type of the object. While the formatting option is still supported, the justification option is no longer supported in the previous context, within the "Display Format" settings. To right/left/center justify text of an object, use the "Format" tool bar which supports many text formatting options.

If you see **[]** listed in your label preview for a label object, you must right click on the object, select "Display Format", and remove the [] characters.

- **Page Setup**
  For DOS to Windows conversions, the default page size setting is set to it's largest values in order to make sure no objects are lost in the conversion process. You must make sure you alter your page settings back to your desired values in order for your labels to print correctly.

  To alter the page size settings, follow the below instructions:

  1. Select "File" > "Page Setup..." from the main Menu Bar.
  2. Select the "Paper Size" tab
  3. For Portrait orientation reports, the Width should be set to "8.5".
  4. For Landscape orientation reports, the Height should be set to "8.5".

     From this dialog, you can also adjust your labels margins from the "Margins" tab.

  After making any changes, make sure that all of your labels objects are located within the defined page size settings and margins. Otherwise, R:BASE will display a warning that "Controls beyond the new page width will be deleted." In this case, select "Cancel" to avoid losing any controls. Then move your controls within your desired page settings and then make your page size setting changes again.

- **Label Preview Tip**
  The "Label Preview" tab will display a preview of your label as it may appear when you send the label to the printer or screen. Before selecting the tab, you must make sure that there are no errors listed in your label variables, otherwise, you will receive an error message for every error listed multiplied by the number of records displayed in the Label Preview window.

  What you can do to limit the number of rows displayed in the Label Preview window, thus limiting the number error messages. To do so, alter the Label Default settings for how many rows are displayed when previewing a label.

  1. Select "Settings" > "Report/Label Designer" > "Default Settings..." from the main Menu Bar.
  2. Alter the value for "Row Count for Preview:" to a smaller value other than 100; perhaps to 10.

- **Lookup Variable Tip**
  Recently added to the R:BASE 11 Label Designer is the option to run a command file (or Action) for a form before the Label Designer is launched. In the Label Designer, choose "Label" from the Menu bar, the select "Actions" > "On Before Design...".

  This allows you to create any variables that Lookup Variables are based upon to avoid annoying error messages when the label is opened in the designer. An example of a lookup variable based upon another variable would be...

```
1 : TEXT    vCompany = Company IN Customer WHERE CustID =
.vCustID
```

  The error message appears because the variable vCustID is not defined. Adding the following SET VAR command to the "On Before Design..." action will alleviate this error message:

```
SET VAR vCustID INTEGER
```

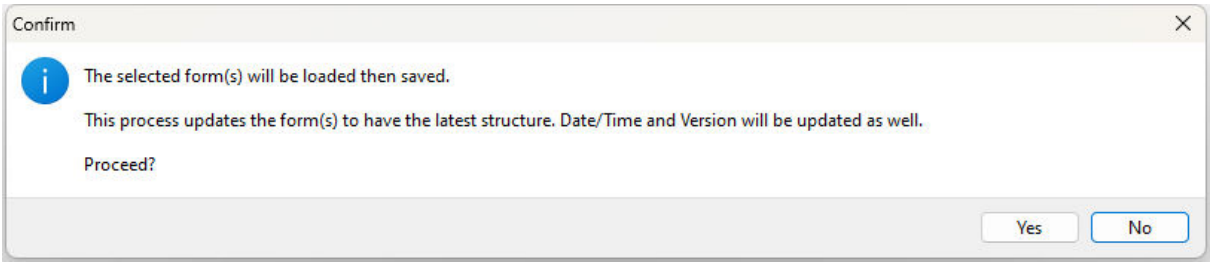
  The "On Before Design..." action is located under "Label" > "Actions" on the main Menu Bar.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.9.3.2 R:BASE 7.x/Turbo V-8 Forms, Reports, & Labels

R:BASE 7.x/Turbo V-8 forms, reports, and labels will automatically populate the Forms, Reports, and Labels menus of the R:BASE 11 Database Explorer. The items will run in R:BASE 11 with little or no changes required on your behalf. However, it is recommended that you review each item to ensure the modules behave as expected.

To assist with the database migration use the "Touch" utility to quickly update and save forms, reports, and labels when migrating between current versions. The Database Explorer right click menu works like opening the selected object in the designer, making the content dirty, and saving it. The feature allows a user to save hundreds of forms, reports, and labels when migrating between R:BASE versions.



Corrections have been made in the form table settings for the "Update Existing Rows" option which was corrected to recognized the setting value. The following command can be used to scans and set all form's "Update Existing Rows" option to checked/enabled. Note that you only need to run the command once.

```
PROPERTY APPLICATION RESET_FORMS_ROW_UPDATE_FLAG '*'
```

A form list may be specified to scan and set the specified form's "Update Existing Rows" option to checked/enabled:

```
PROPERTY APPLICATION RESET_FORMS_ROW_UPDATE_FLAG 'form1,form2,form3'
```

Leaving the last parameter blank scans and sets all form's "Update Existing Rows" option to checked/enabled, where the form modified date is less than November 7, 2017, which is when the "Update Existing Rows" option has been corrected to recognized the setting value.

```
PROPERTY APPLICATION RESET_FORMS_ROW_UPDATE_FLAG ' '
```

To assist in the migration to R:BASE 11, new conversion features have been built in for your convenience.

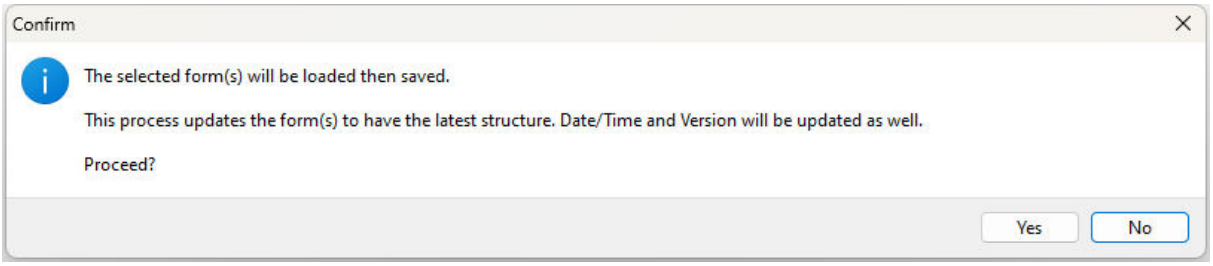
Please review the "What's New in R:BASE 11 for Windows" PDF document for additional details.

1.9.3.3 R:BASE eXtreme 9.x (32/64) Forms, Reports, & Labels

Your R:BASE eXtreme 9.x Forms, Reports, and Labels will automatically populate the Forms, Reports, and Labels menus of the R:BASE 11 Database Explorer. The items will run in R:BASE 11 with no changes required on your behalf. It is recommended that you review each item to ensure the modules behave as expected.

To assist with the database migration use the "Touch" utility to quickly update and save forms, reports, and labels when migrating between current versions. The Database Explorer right click menu works like opening the selected object in the designer, making the content dirty, and saving it. The feature allows a user to save hundreds of forms, reports, and labels when migrating between R:BASE versions.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Corrections have been made in the form table settings for the "Update Existing Rows" option which was corrected to recognized the setting value. The following command can be used to scans and set all form's "Update Existing Rows" option to checked/enabled. Note that you only need to run the command once.

```
PROPERTY APPLICATION RESET_FORMS_ROW_UPDATE_FLAG '*'
```

A form list may be specified to scan and set the specified form's "Update Existing Rows" option to checked/enabled:

```
PROPERTY APPLICATION RESET_FORMS_ROW_UPDATE_FLAG 'form1,form2,form3'
```

Leaving the last parameter blank scans and sets all form's "Update Existing Rows" option to checked/enabled, where the form modified date is less than November 7, 2017, which is when the "Update Existing Rows" option has been corrected to recognized the setting value.

```
PROPERTY APPLICATION RESET_FORMS_ROW_UPDATE_FLAG ' '
```

To assist in the migration to R:BASE 11, new conversion features have been built in for your convenience.

Please review the "What's New in R:BASE 11 for Windows" PDF document for additional details.

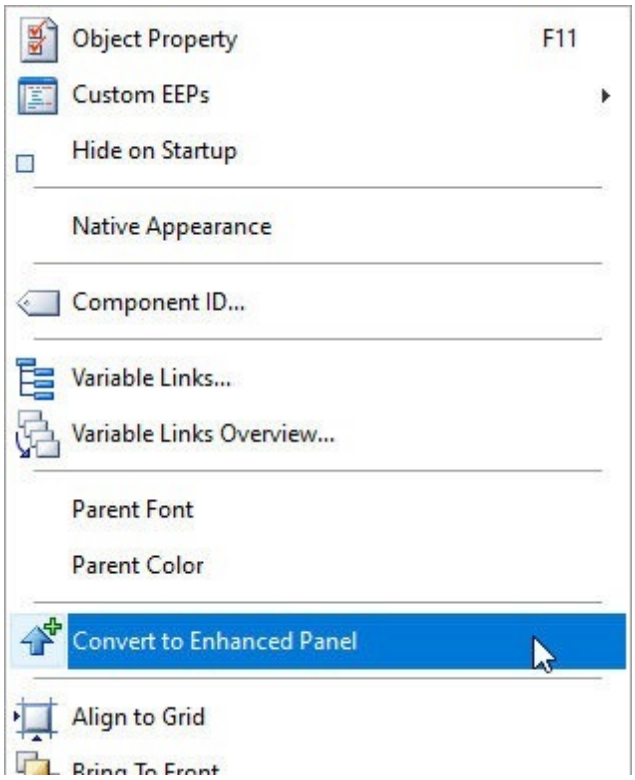
1.9.3.4 New Conversion Enhancements

To assist in the migration to R:BASE 11, new conversion features have been built in for your convenience.

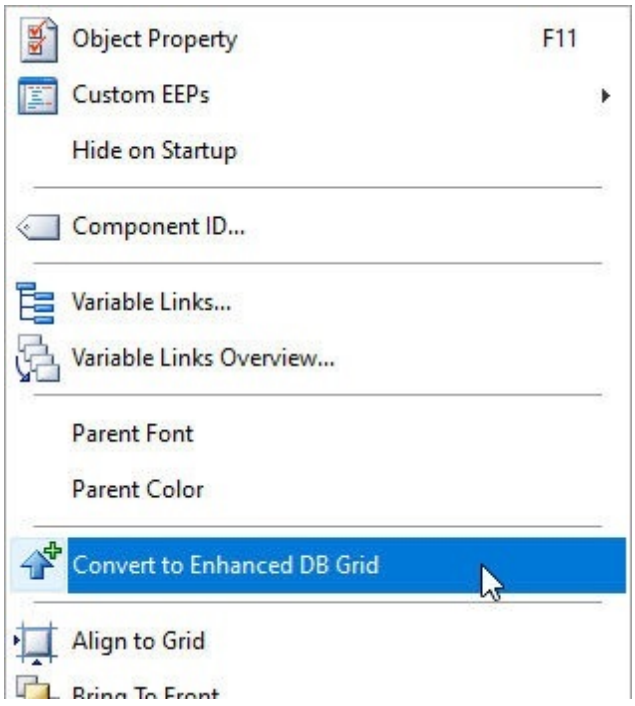
"Single Click" Conversion to Enhanced Form Controls
In the earlier versions of R:BASE, there are several introductory controls that have been enhanced considerably in later R:BASE releases. For example, the Tab Control was provided in R:BASE 7.0 for Windows. Later, in R:BASE 7.1 the Enhanced Tab Control was introduced to offer many additional features. The same comparison can be made for the popular DB Grid control whose "enhanced" counterpart was introduced in R:BASE eXtreme for Windows. Migrating from the introductory control to the enhanced counterpart meant recreating the object on the form, while referencing the properties of the original.

Now, the conversion of the introductory control can be made with a single click. For example, by right clicking on a Panel control, the option to "Convert to Enhanced Panel" is available.

Copyright © 1982-2024 R:BASE Technologies, Inc.



By right clicking on a DB Grid control, the option to "Convert to Enhanced DB Grid" is available.

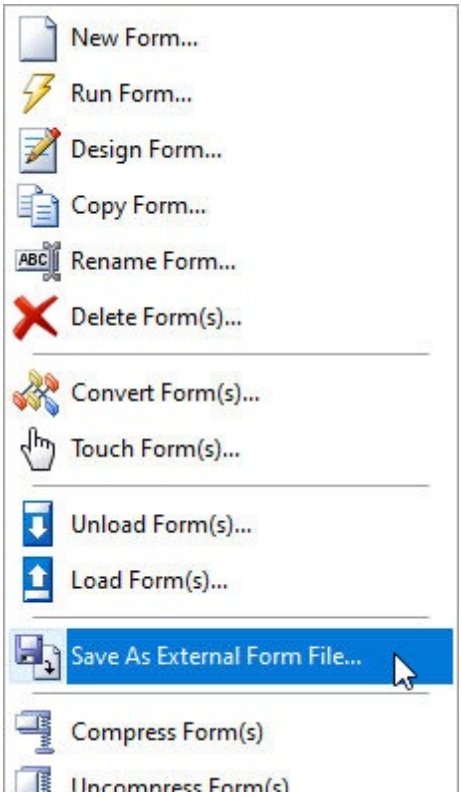


The following single-click conversions to "enhanced" form controls are available:

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Panel -> Convert to Enhanced Panel
- Group Box -> Convert to Enhanced Group Box
- Speed Button -> Convert to Enhanced Speed Button
- Tab Control -> Convert to Enhanced Tab Control
- Calendar -> Convert to Enhanced Calendar
- Variable Calendar -> Convert to Enhanced Variable Calendar
- DB Calendar -> Convert to Enhanced DB Calendar
- DB Grid -> Convert to Enhanced DB Grid
- DB Navigator > Convert to Enhanced DB Navigator
- DB Rich Edit -> Convert to Advanced DB Rich Edit
- Speed Button -> Convert to Enhanced Speed Button

Variable Form Conversion to External Form File
Another new time-saving conversion feature is located in the Database Explorer within the "Forms" options where a new selection allows for a Variable Form to be saved as an External Form File. The option is only enabled when a Variable Form is selected.



Please review the "What's New in R:BASE 11 for Windows" PDF document for additional features.

1.9.4 Converting Applications

As R:BASE can be customized for any type of business model or company schema, there have always been several ways that developers and users could create an application. Actually, if you think of what an application consists of, you're really looking at a stored series of commands in what are called command files, which are then combined with other command files to form applications.

R:BASE provides all of the tools you need to produce command files and applications, it is just up to you to decide which method serves your needs best.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.9.4.1 R:BASE 6.5++ and Lower Applications

1.9.4.1.1 Application Express Users (.APP)

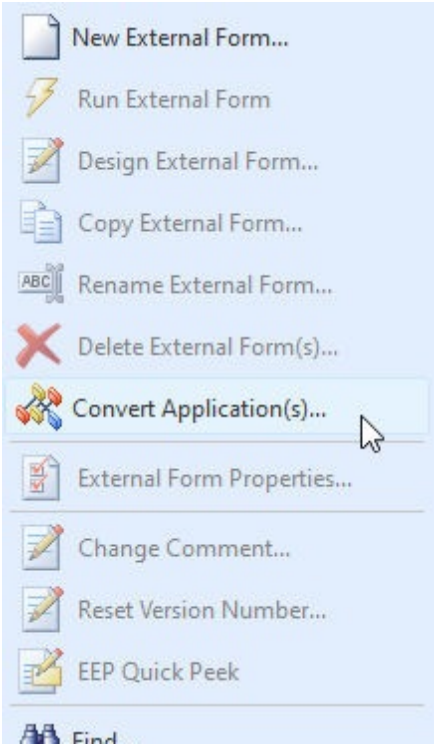
If you are converting an application that was designed using the Application Express module from versions prior to R:BASE 7.x, the actual application file consisting of the R:BASE command language (.APP) can be converted within R:BASE 11. In addition to the .APP file, there will be an .APX and either .API or .APW files. You will only need the .APP or the .API files.

The application may be one large .APP file or several smaller command files; usually with the .CMD file extension. The standard command file extension was .CMD in early R:BASE versions. It may also be possible that no file extension is used for the command files. In this case, it is recommended that the .RMD file extension be added to the files so it may be displayed within the "Command Files" menu of the Database Explorer's Navigator.

In R:BASE for Windows, the new file extension standard is .RMD, as more file servers and mail servers flag the .CMD file extension as a possible virus. Or, if you are using a custom file extension, you can add the extension to the "File Mask:" field under the "Command Files" menu, so the files will then appear in this window.

The source code command syntax within the .APP file contains the menus and menu actions of your custom application in the form of command blocks, menu blocks, and possibly screen blocks. The .APP file is likely located with your database files. So, when you connect to your database using R:BASE 11, you will see the .APP file listed under the "Command Files" menu within the Database Explorer's Navigator. The files will not be listed under the "Applications" menu as the R:BASE 11 Application Designer will not open legacy Application Express projects for editing.

In order to use the .APP file contents in R:BASE 11, a new "**Convert Application(s)...**" option has been made available within the "External Forms" portion of the Database Explorer. This feature will parse through the legacy Windows and DOS legacy application files and converts it to an External Form File. If the file does not contain $COMMAND blocks or $MENU Blocks, then no conversion will take place.



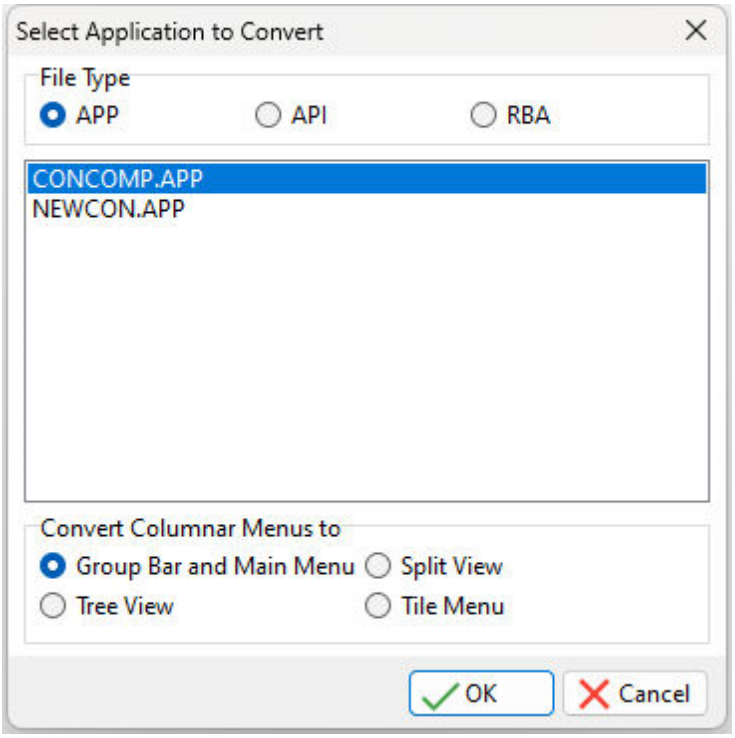
Selecting the "Convert Application(s)..." option will display a dialog to select an application file (.APP/.API). If no files are displayed in the dialog, press the "Cancel" button and select "Utilities" > "Set

Copyright © 1982-2024 R:BASE Technologies, Inc.

Working Directory..." to navigate to the folder where the application files reside. Then, revisit the "Convert Application(s)..." option.

1.9.4.1.1.1 Menu System Types

The converted external form will vary based upon the menu type (POPUP, COLUMN, or PULLDOWN) of the .APP application file. With the Tree View, Split View, and Tile Menu application conversions, the COLUMN and PULLDOWN items are merged, and POPUP items become form commands. With the "Group Bar and Menu Bar" conversion, COLUMN items becomes a Group Bar, PULLDOWN items becomes a Main Menu, and POPUP items become form commands. All menu types will create a collection of user interface groups that contains a list of items to select that can invoke commands.



After selecting the "OK" button, the converted external form will appear in the Database Explorer window. An external form file, with the .RFF file extension, will reside in the folder. If the database and application is moved to another directory, be sure to move the external form file as well. To design the external form and see the converted command syntax, select "Design External Form..." from Navigator options. You will then enter the External Form Designer module to alter the external form.

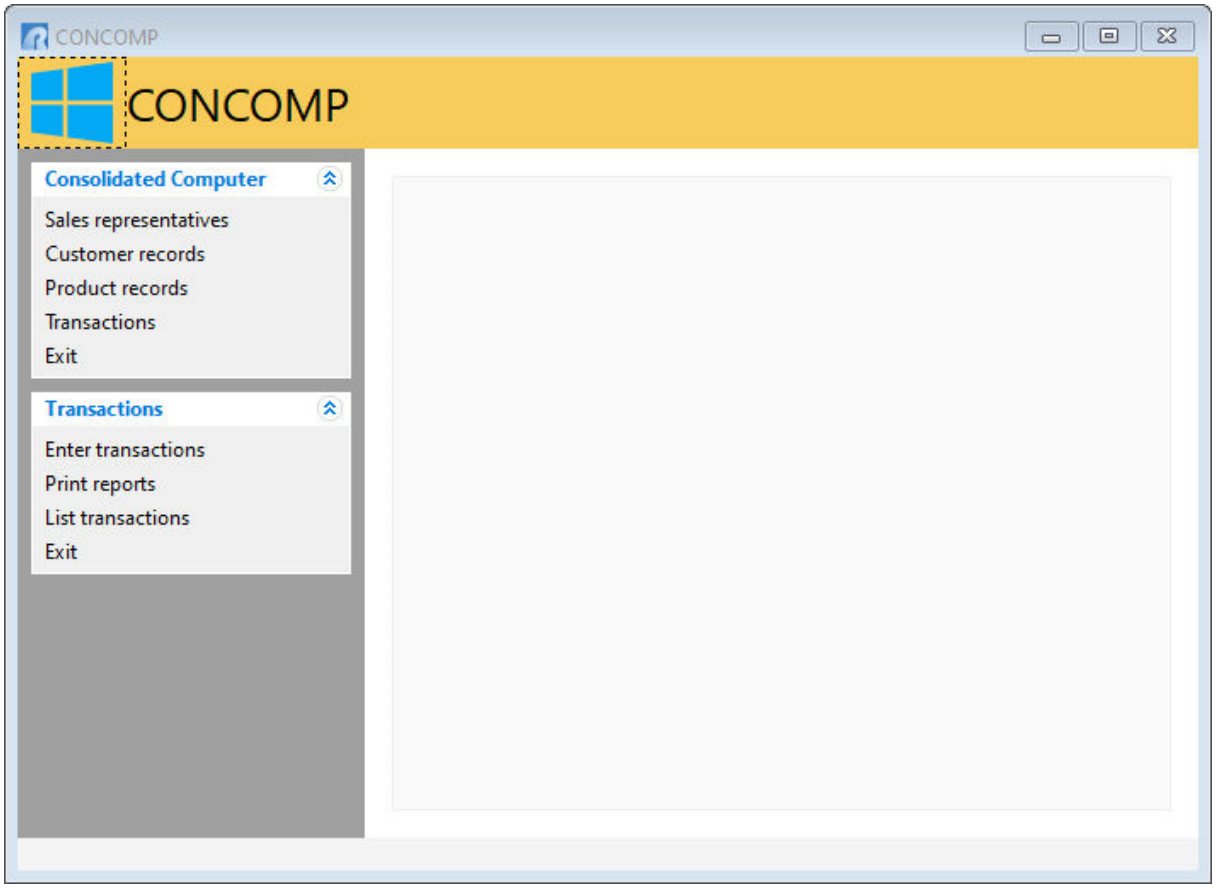
Based upon the menu system type of the original .APP application file, R:BASE will create the following menu system types into the external form:

- Group Bar
- Menu Bar
- Tree View
- Split View
- Tile Menu

After the conversion, the Group Bar will be located on the left side of the form with menu options placed accordingly. The Group Bar items displayed should match the $MENU blocks from the originating .APP application file. There may be some menu options that are no longer used. Make a note of the items as this will save time in the process of reviewing and updating the command syntax.

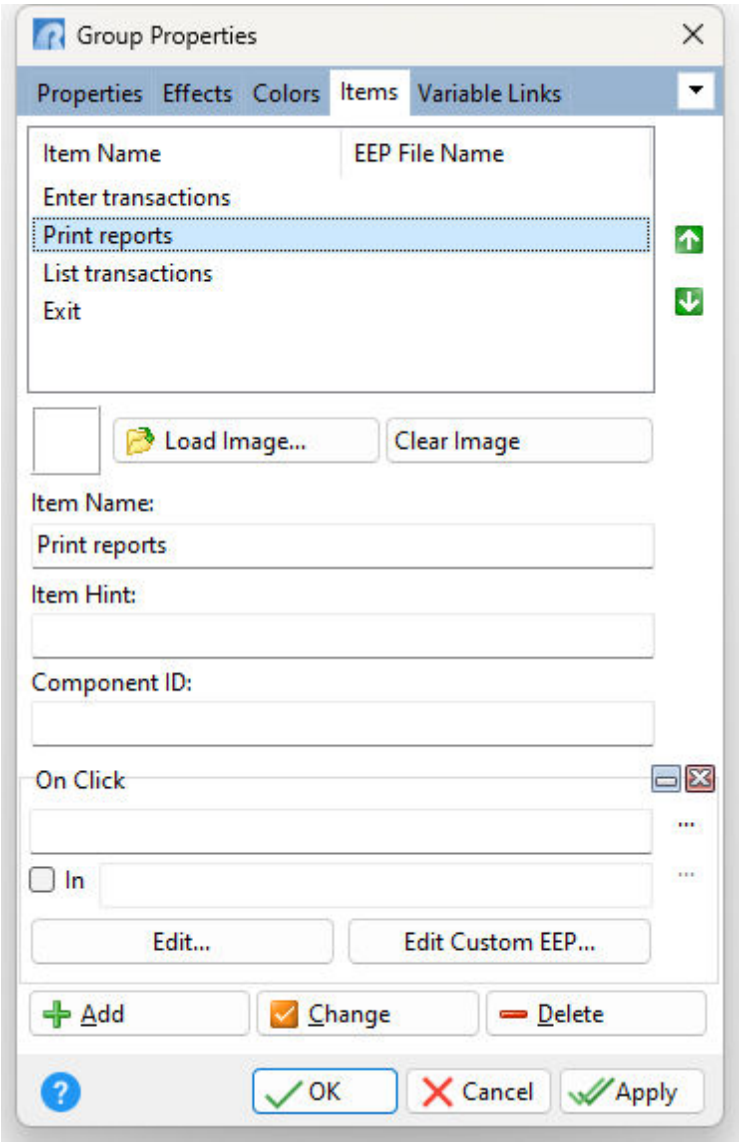
Copyright © 1982-2024 R:BASE Technologies, Inc.

A Group Bar control is a collection of "groups", that can be opened and closed by using the arrows (looks like two carat characters ^). Each group contains a list if items that are used to run a series of R:BASE commands or an external file. The "Groups" are displayed below with a blue font. The Group Items are located under the Group with a black font. In the below example image, "Transactions" is the second Group, where "Enter transactions", "Print reports", "List transactions", and "Exit" are the Items.



By right clicking on the Group, and selecting "Object Property [F11], the Group Properties dialog is displayed. Select the "Items" tab to review the list of item names. Any command syntax loaded in an Item will highlight the "Edit Custom EEP..." button with a yellow background. Selecting the "Edit Custom EEP..." button will display the stored command syntax. Before running the external form and selecting any menu options, the command syntax must first be checked. If no syntax was loaded into the EEP, then the button background will not be yellow. The command syntax would then be loaded into the Custom Form Actions. Select "Layout" > "Custom Form Actions" from the main menu bar to review the command syntax.

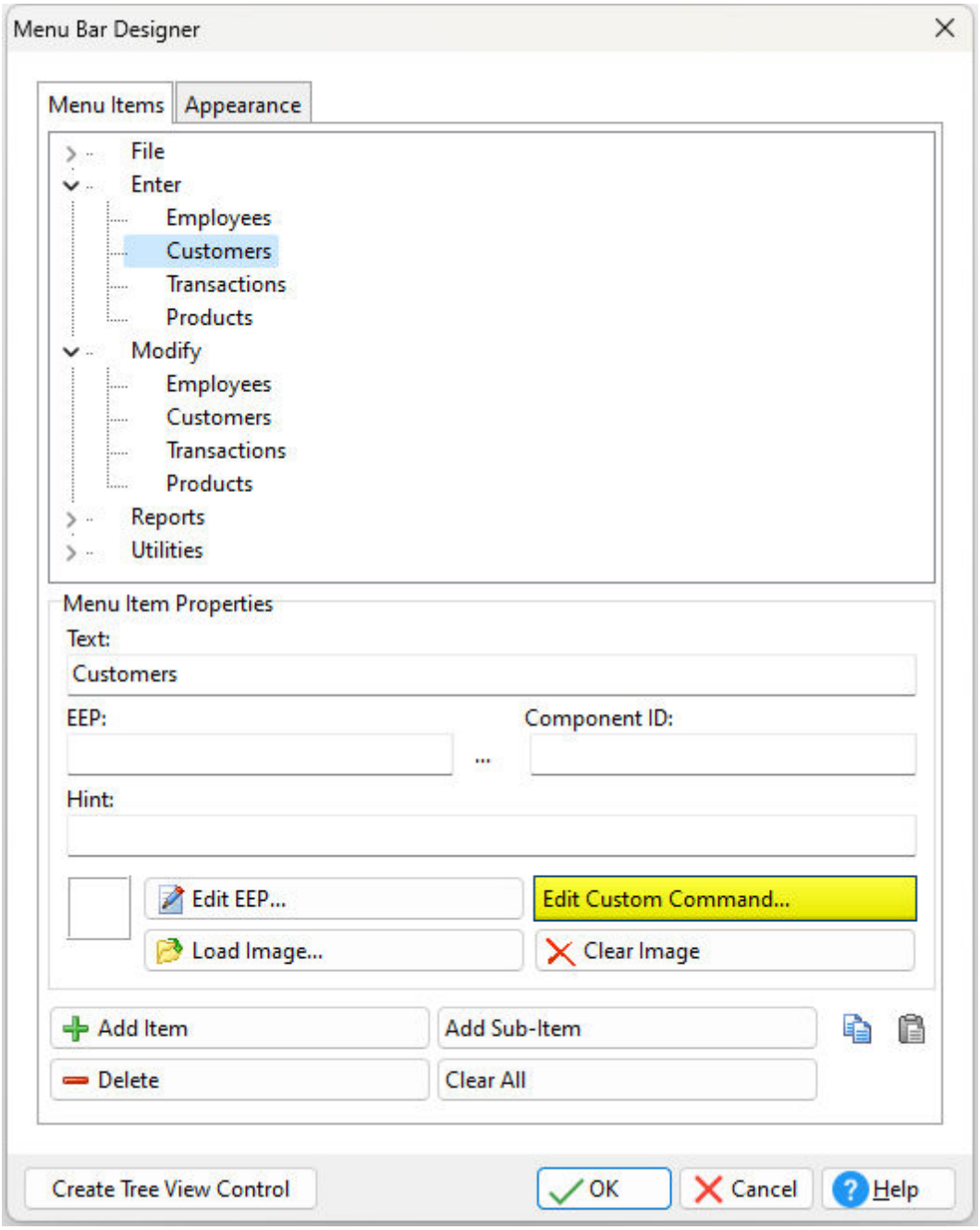
Copyright © 1982-2024 R:BASE Technologies, Inc.



After the conversion, a blank gray canvas will be displayed in the External Form Designer. The menu system and command syntax for each menu item from the application are stored within the external form file. From the main Menu Bar, select "Layout" > "Design Menu Bar". The Menu Bar Designer will launch with the application file menus placed accordingly. The Menu Items displayed should match the $MENU blocks from the originating .APP application file. There may be some menu options that are no longer used. Make a note of the items as this will save time in the process of reviewing and updating the command syntax.

A Menu Bar is a horizontal strip that contains lists of available menus for a certain program. In Windows programs, the menu bar resides at the top of a window. The Menu Bar uses main "Items" which display across the top of a window, and "Sub-Items" which drop down when the main Item is selected. Items" and Sub-Items can be used to run a series of R:BASE commands or an external file. In the below example, "Employees" is the main Item, where "Enter New Employee" and "Edit Existing Employee" are the sub-items.

Copyright © 1982-2024 R:BASE Technologies, Inc.



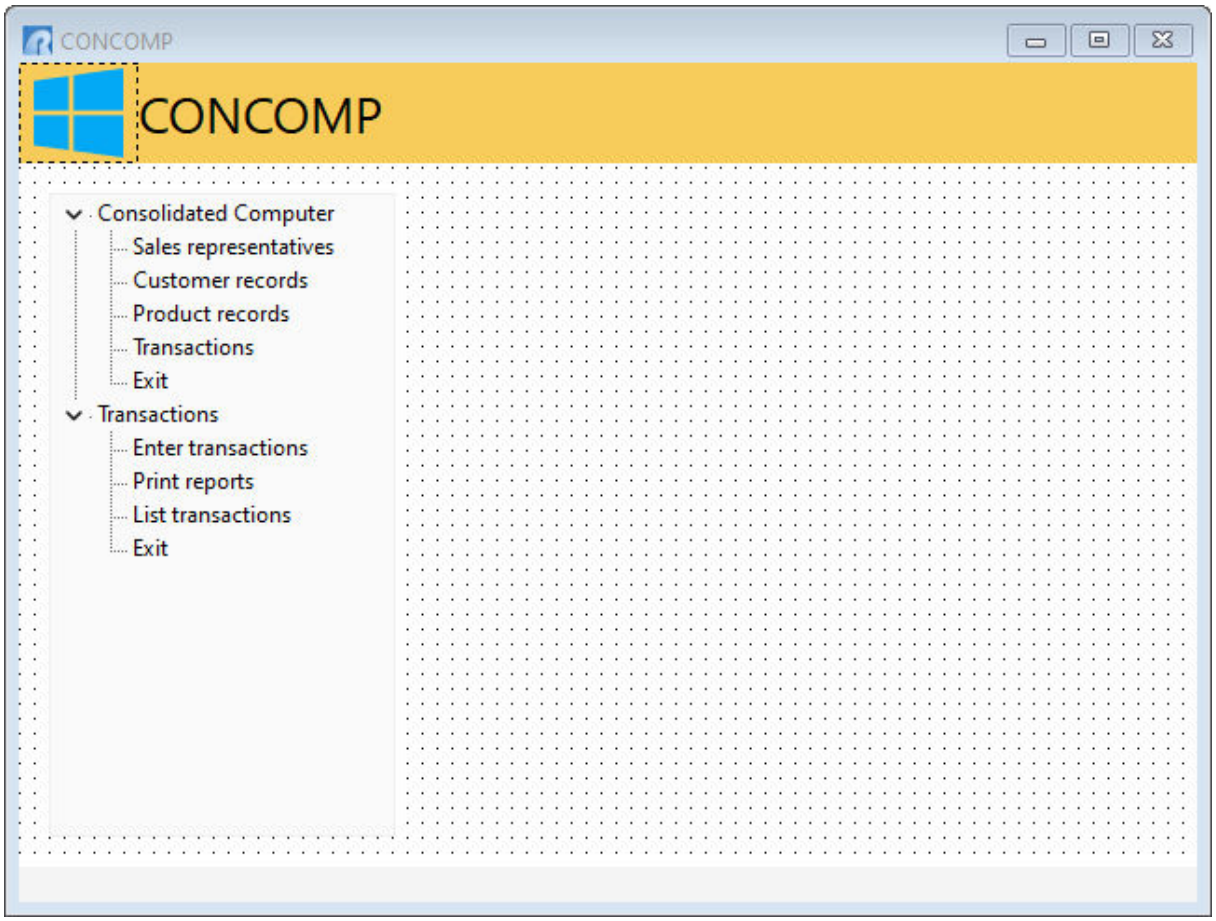
During the conversion, R:BASE loaded the command syntax portions of the .APP application file into the "Custom Command" portions of the Menu Bar. Any command syntax loaded in a Sub-Item will highlight the "Edit Custom Command..." button with a yellow background. Selecting the "Edit Custom Command..." button will display the command syntax stored within the Item or Sub-Item. Before running the external form and selecting any menu options, the command syntax must first be checked.

If no syntax was loaded into the menu item or sub-item, then the button background will not be yellow. The command syntax would then be loaded into the Custom Form Actions. Select "Layout" > "Custom Form Actions" from the main menu bar to review the command syntax.

Copyright © 1982-2024 R:BASE Technologies, Inc.

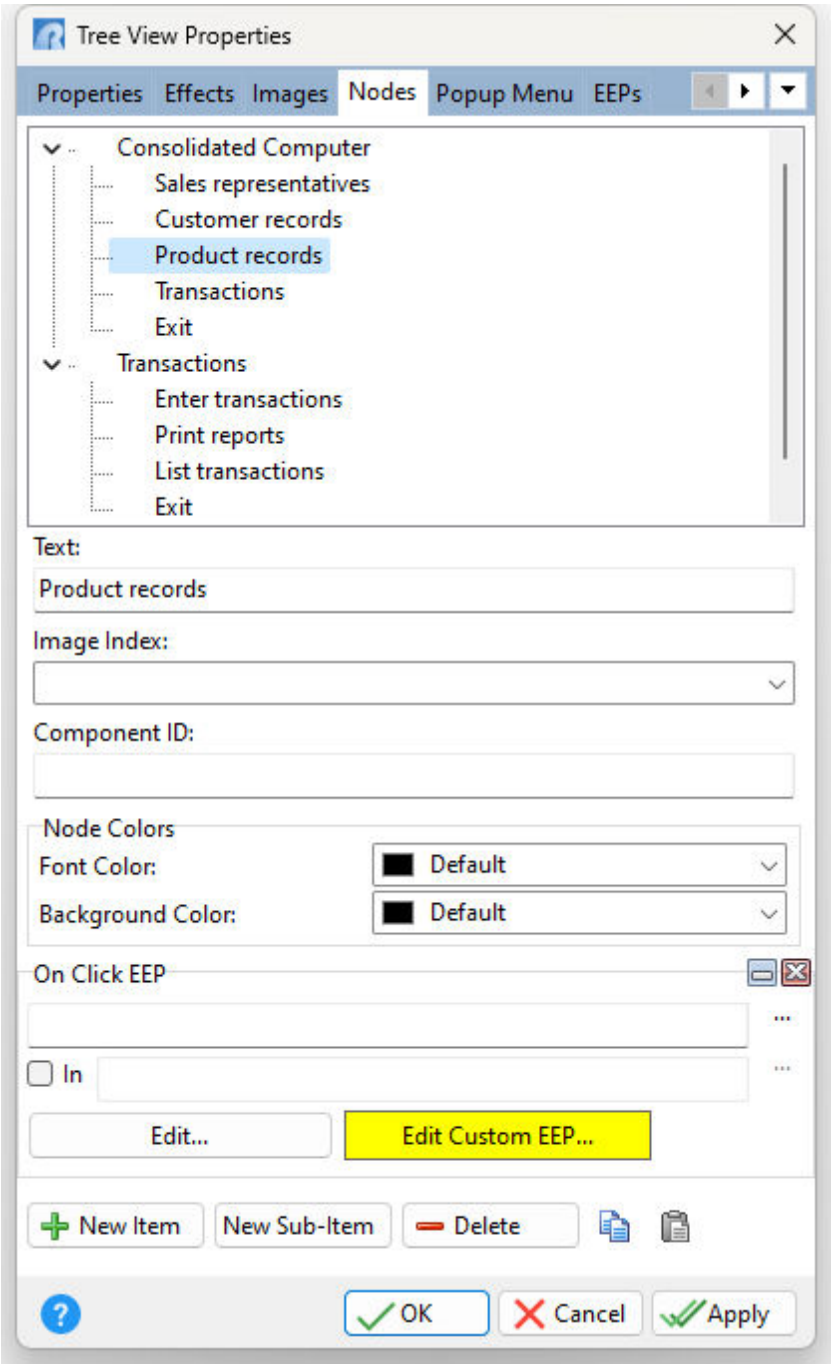
After the conversion, the Tree View will be located on the left side of the form with menu options placed accordingly. The Tree View items displayed should match the $MENU blocks from the originating .APP application file. There may be some menu options that are no longer used. Make a note of the items as this will save time in the process of reviewing and updating the command syntax.

A Tree View control displays an object containing a hierarchical list of static items (nodes), such as the headings in a document, the entries in an index, or the files and directories on a disk. Each node consists of a label and an optional image, where each node can have a list of subitems associated with it. By selecting the tree structure, the user can expand or collapse the associated list of subitems. The main "Nodes" are displayed below next to the minus characters. The Sub-Items are located within the Nodes linked by the dotted lines. In the below example image, "Transactions" is the second node, where "Enter transactions", "Print reports", "List transactions", and "Exit" are the sub-items.



By right clicking on the Tree View, and selecting "Object Property [F11], the Tree View Properties dialog is displayed. Select the "Nodes" tab to review the list of nodes and sub-items. Any command syntax loaded in an sub-item will highlight the "Edit Custom EEP..." button with a yellow background. If no syntax was loaded into the sub-item, then the button background will not be yellow. Selecting the "Edit Custom EEP..." button will display the stored command syntax. Before running the external form and selecting any menu options, the command syntax must first be checked. If no syntax was loaded into the EEP, then the button background will not be yellow. The command syntax would then be loaded into the Custom Form Actions. Select "Layout" > "Custom Form Actions" from the main menu bar to review the command syntax.

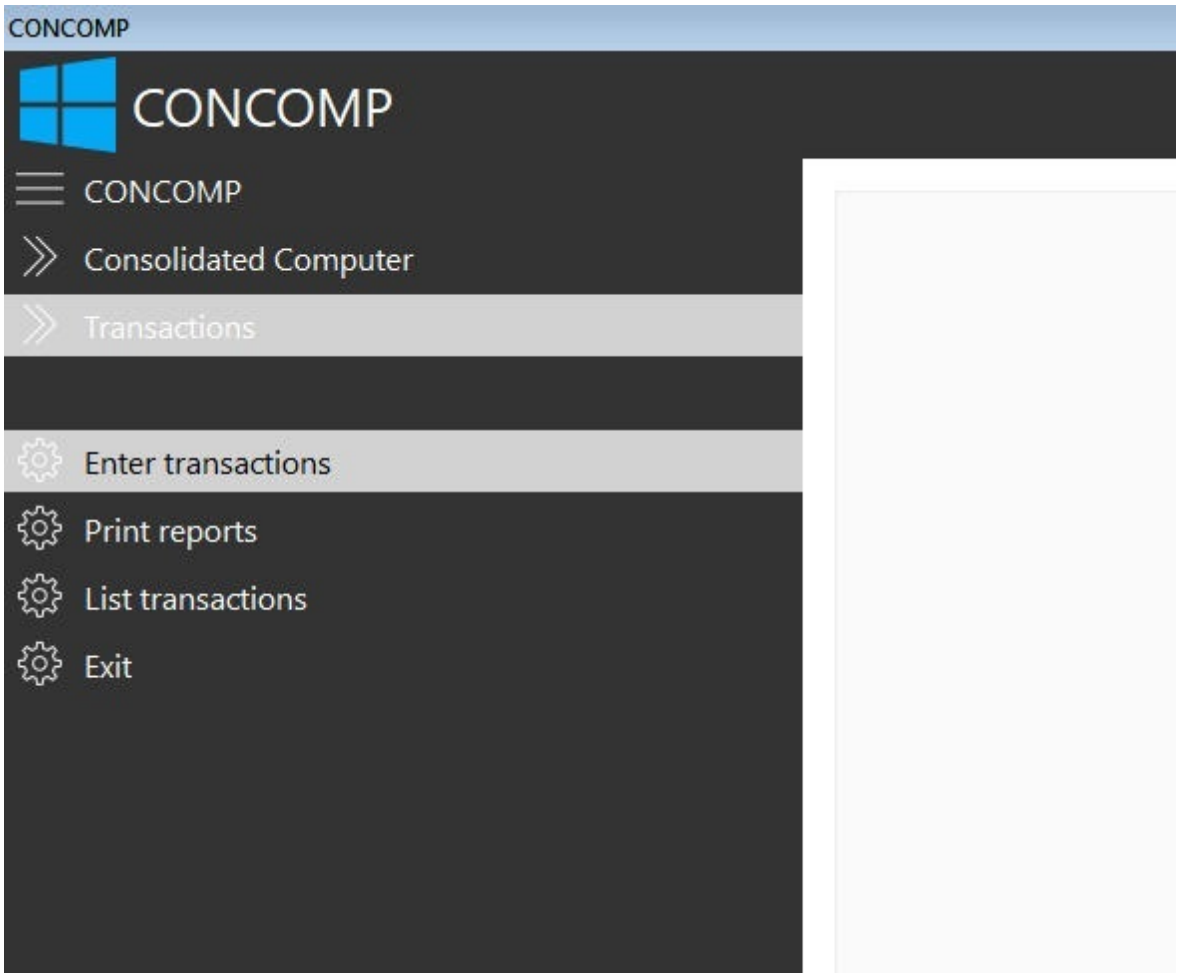
Copyright © 1982-2024 R:BASE Technologies, Inc.



After the conversion, the Split View will be located on the left side of the form with menu options placed accordingly. The Split View items displayed should match the $MENU blocks from the originating .APP application file. There may be some menu options that are no longer used. Make a note of the items as this will save time in the process of reviewing and updating the command syntax.

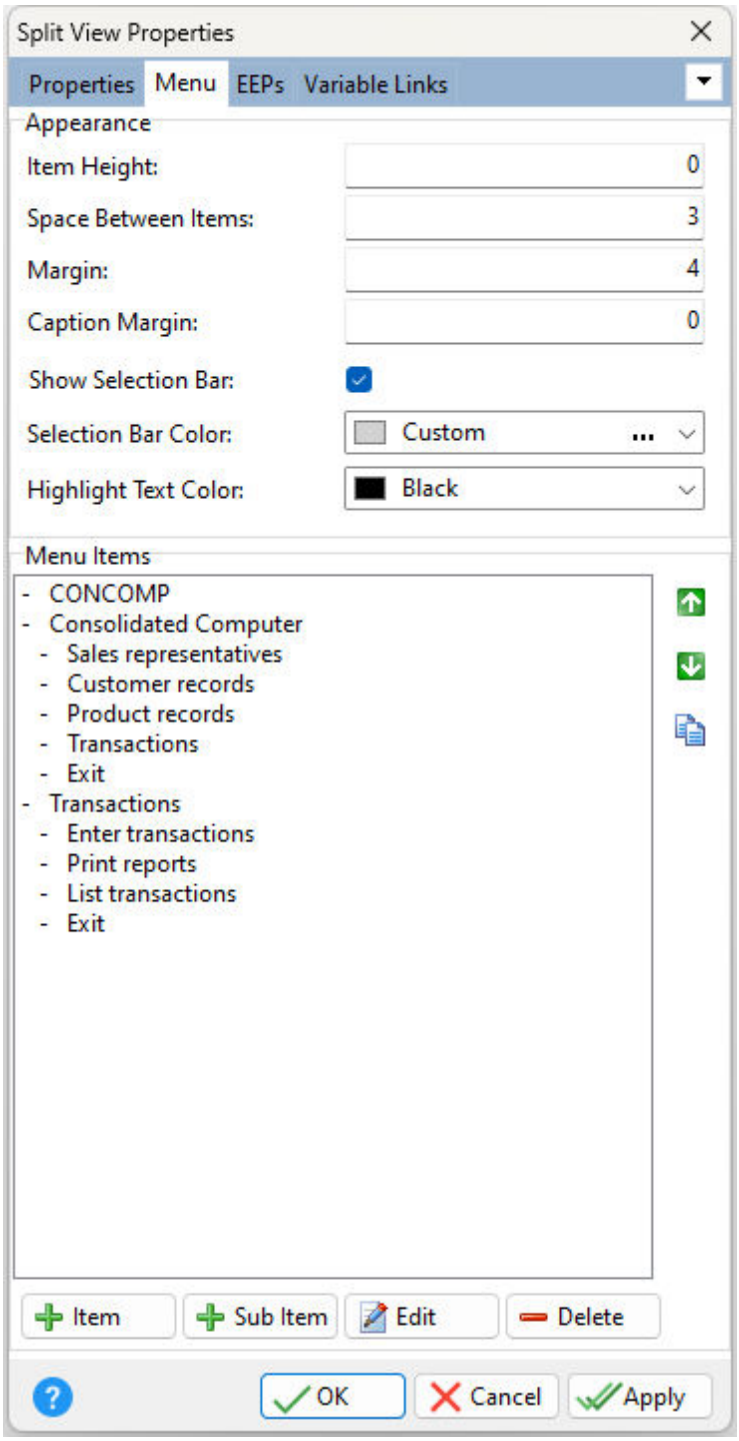
The Split View is a control offering a collection of user interface command groups. A command group typically contains a list of clickable items that invoke commands. The Split View presents a list of initial menu options, which is then expanded when the mouse cursor hovers over the list. In the below example image, "Transactions" is the main menu item, where "Enter transactions", "Print reports", "List transactions", and "Exit" are the sub items.

Copyright © 1982-2024 R:BASE Technologies, Inc.



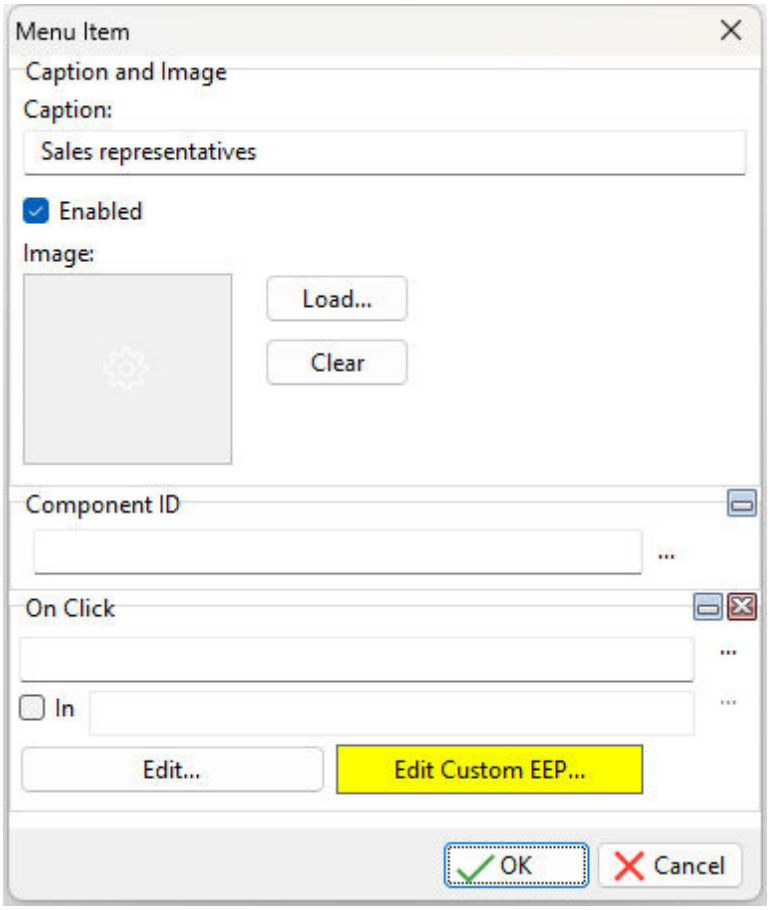
By right clicking on the Split View, and selecting "Object Property [F11], the Split View Properties dialog is displayed. Select the "Menu" tab to review the list of menu items.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Select a menu item and then the "Edit" button to review the menu item's properties. Any command syntax loaded in a menu Item will highlight the "Edit Custom EEP..." button with a yellow background. Selecting the "Edit Custom EEP..." button will display the stored command syntax. Before running the external form and selecting any menu options, the command syntax must first be checked. If no syntax was loaded into the EEP, then the button background will not be yellow. The command syntax would then be loaded into the Custom Form Actions. Select "Layout" > "Custom Form Actions" from the main menu bar to review the command syntax.

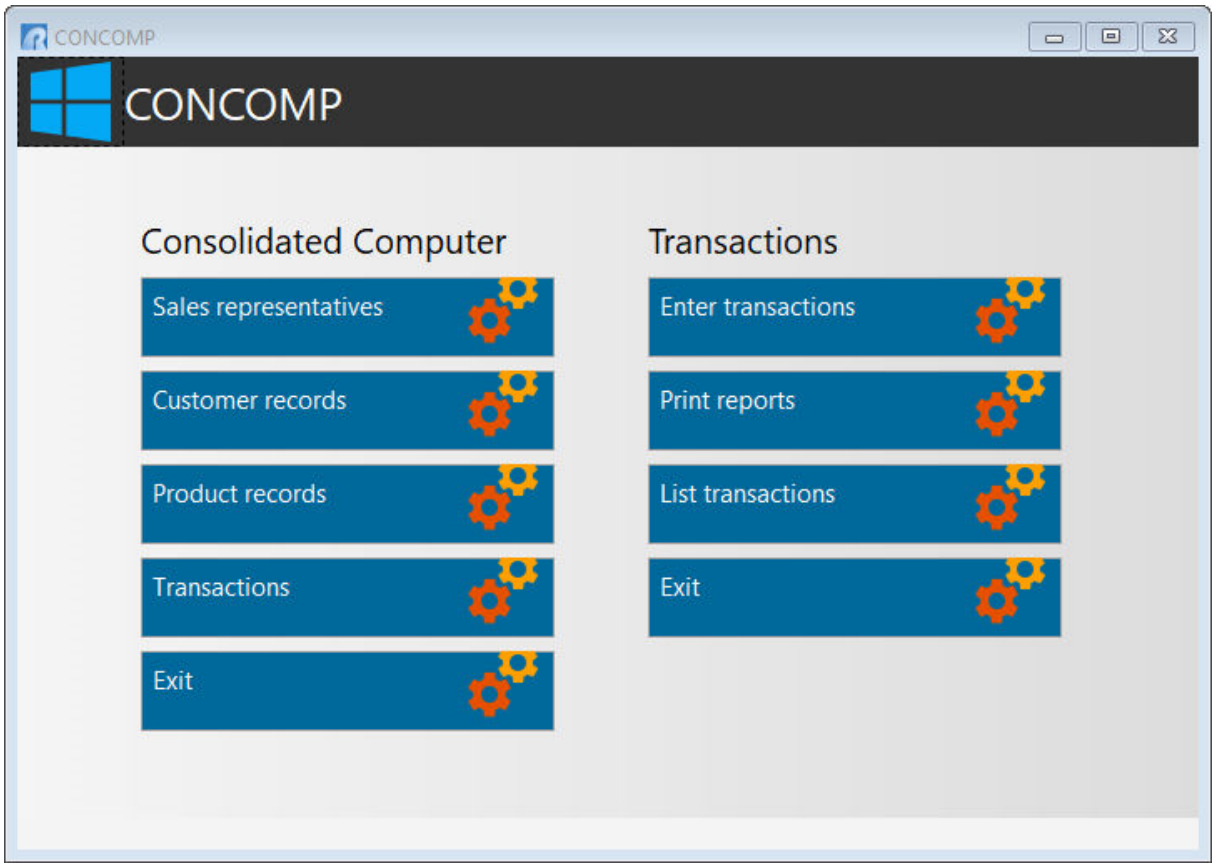
Copyright © 1982-2024 R:BASE Technologies, Inc.



After the conversion, the Tile Menu will be located on the center of the form with menu options placed accordingly. The Tile Menu items displayed should match the $MENU blocks from the originating .APP application file. There may be some menu options that are no longer used. Make a note of the items as this will save time in the process of reviewing and updating the command syntax.

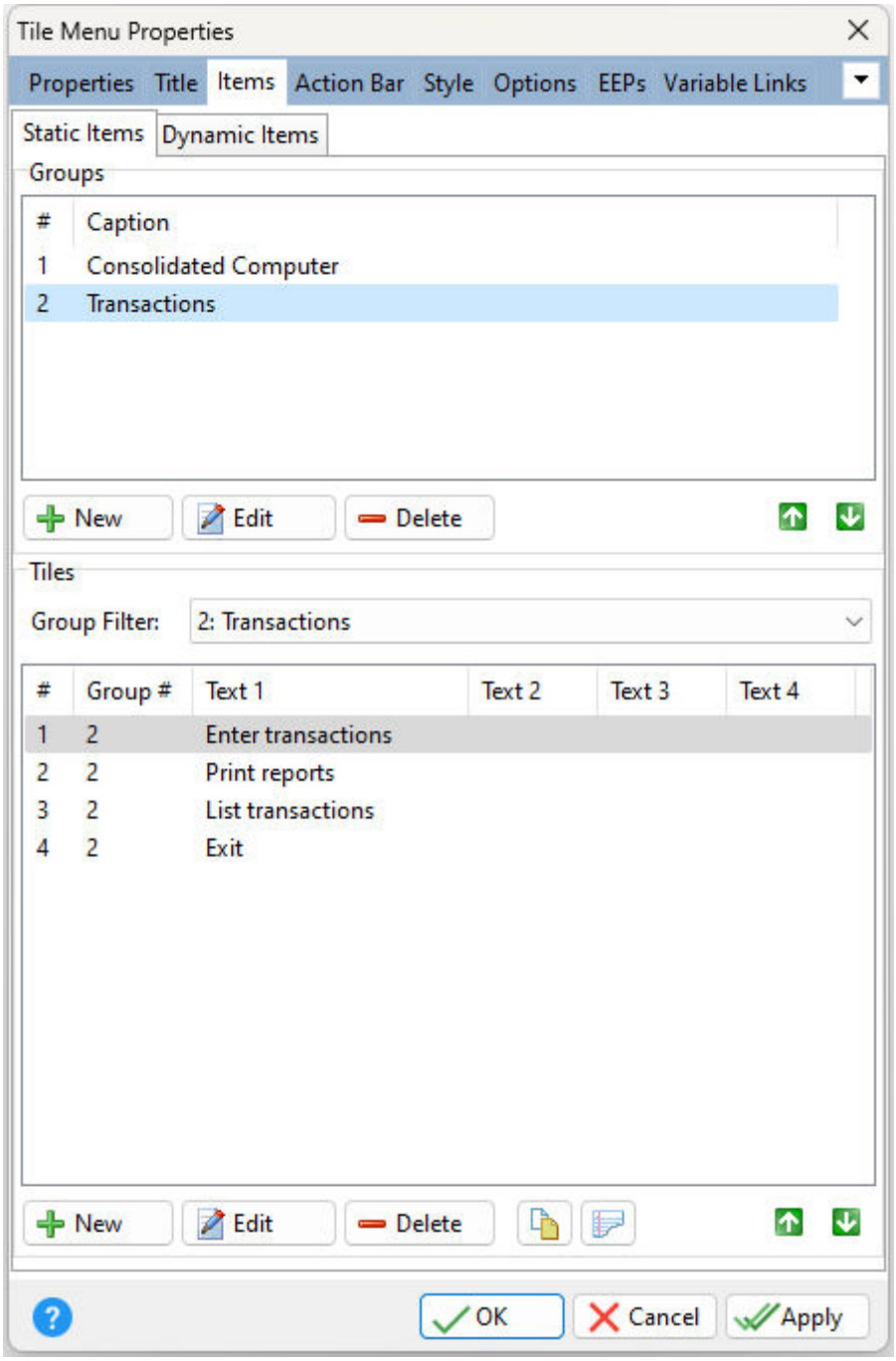
The Tile Menu is a control offering a tile (colored rectangles or squares) interface to create a collection of user interface command groups. A command group typically contains a list of clickable items that invoke commands. The Tile Menu is both a menu control and a container, where the tile and tile group functionality can be used to create a powerful dashboard for an R:BASE application. In the below example image, "Transactions" is a main group, where "Enter transactions", "Print reports", "List transactions", and "Exit" are the tiles.

Copyright © 1982-2024 R:BASE Technologies, Inc.



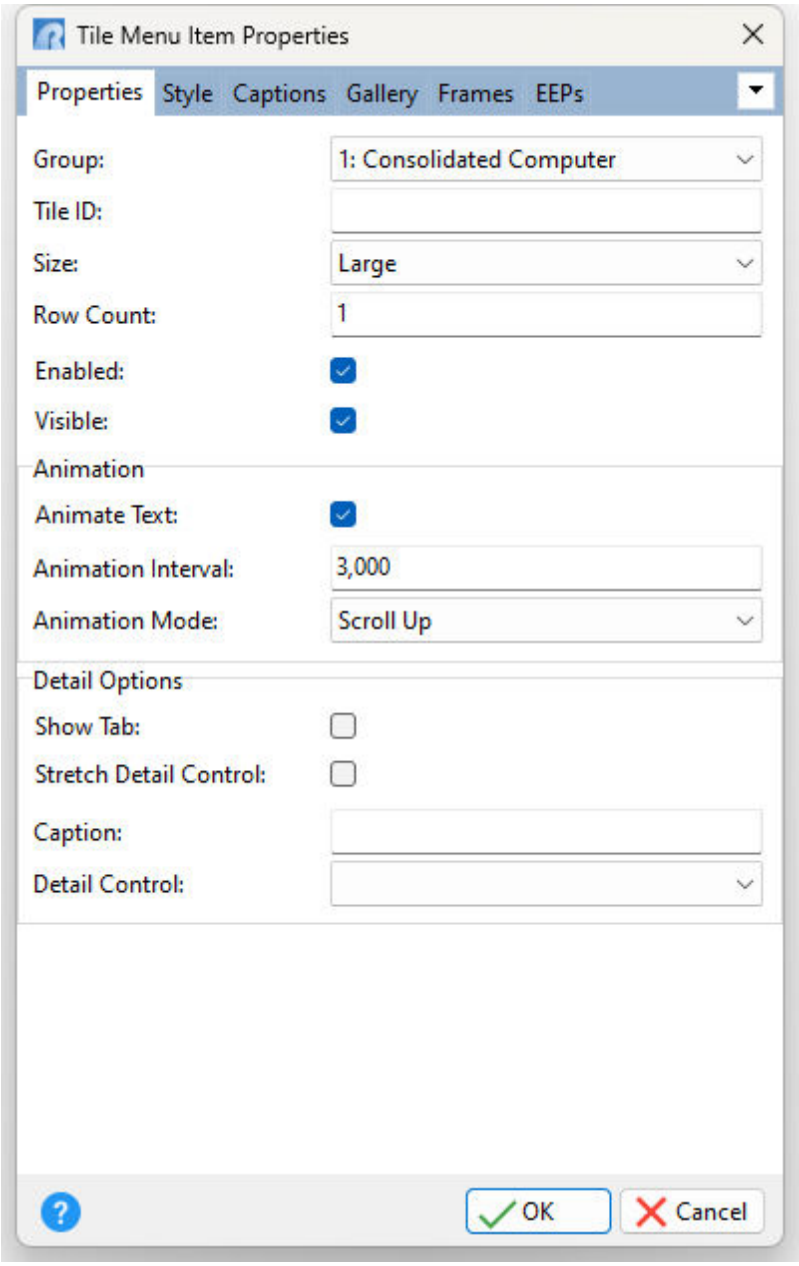
By right clicking on the Tile Menu, and selecting "Object Property [F11], the Tile Menu Properties dialog is displayed. Select the "Items" tab to review the list of group and tile items.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Select a tile item and then the "Edit" button to review the tile item's properties. Under the "EEPs" tab, any command syntax loaded in a tile menu item will highlight the "Edit Custom EEP..." button with a yellow background. Selecting the "Edit Custom EEP..." button will display the stored command syntax. Before running the external form and selecting any menu options, the command syntax must first be checked. If no syntax was loaded into the EEP, then the button background will not be yellow. The command syntax would then be loaded into the Custom Form Actions. Select "Layout" > "Custom Form Actions" from the main menu bar to review the command syntax.

Copyright © 1982-2024 R:BASE Technologies, Inc.



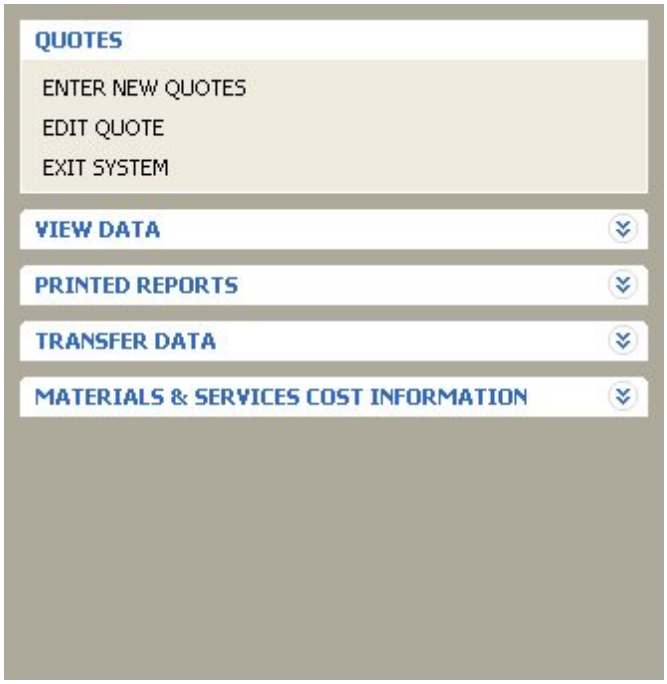
1.9.4.1.1.2 Menu Action Types

When a menu option is selected in a legacy application, R:BASE would have provided another menu (e.g. sub menu) to the user, or would have performed an action. The two type of menus that can be used include a "menu to a menu" or a "menu to an action."

Menu to a Menu
In some cases, a legacy menu selection was used to jump to another menu. This usually occurs in the first or second tier of a menu system. In the menu system below, the menu options with the red, orange, green, and blue lines were used in a DOS menu system to jump to another screen with additional options. The other three options "ENTER NEW QUOTES", "EDIT QUOTE", and "EXIT QUOTE FILES" do not.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Since the Windows menu system can display more information, these initial tier options with the red, orange, green, and blue lines can be removed, and the displayed Group Bar main group captions can be altered. The end result may look something like this.



Menu to an Action
The menu to an action will run a series of commands, a Command Block, or an external file. The action can contain as little as one or two commands, or as much as hundreds of lines of code.

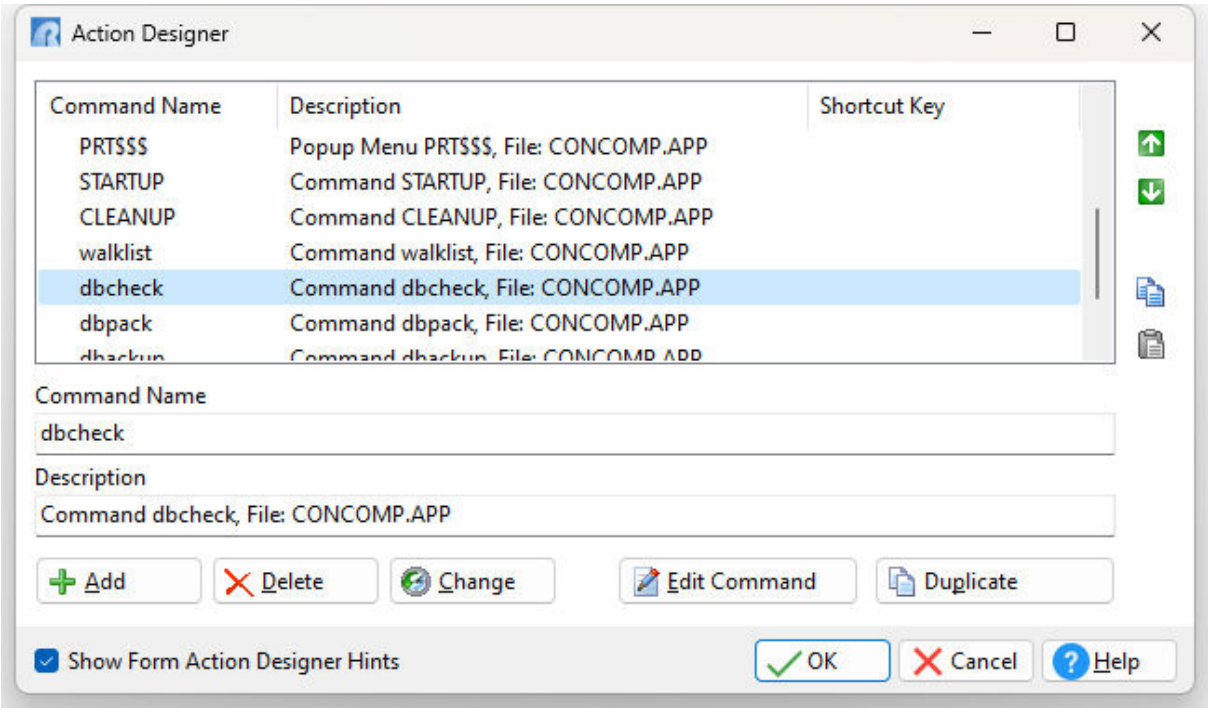
Copyright © 1982-2024 R:BASE Technologies, Inc.

After the conversion, R:BASE loaded the command syntax portions of the .APP application file into the "Custom Command" or "Custom EEP" portions of the new menu system (e.g. Menu Bar, Group Bar, Tree View). Any command syntax loaded will highlight the "Edit Custom Command..." or "Edit Custom EEP" button with a yellow background. Selecting the button will display the stored command syntax. Before running the external form and selecting any menu options, the command syntax must first be checked.

1.9.4.1.1.3 Command Syntax Reference

After the conversion, R:BASE will also load the command block portions of the .APP application file into the external form as separate "Custom Form Actions". From the main Menu Bar, select "Layout" > "Custom Form Actions". The placement of the command syntax within the Custom Form Actions serves as a secondary reference for the command syntax that was used in the legacy application file. If the individual command blocks were already loaded into the appropriate menu system (Menu Bar, Group Bar, Tree View, Split View, Tile Menu) during the initial conversion process, the syntax here does not need to be updated in addition to to the command syntax in the menu system.

The Action Designer, which is used to build and store a library of command blocks, will launch with the application file $COMMAND blocks loaded separately as individual "Command Name" actions. The "Description" column will list the $COMMAND block and originating application file. Within each of the listed items is a block of code. Select or hover the mouse cursor over an item will display a hint of the stored command syntax.



Select the "Edit Command" button to review the command syntax stored within the Custom Form Action.

1.9.4.1.2 Command Files

The application may be one large .APP file or several smaller files; usually with the .CMD file extension. The standard command file extension was .CMD in early R:BASE versions. It may also be possible that no file extension is used for the command files. In this case, it is recommended that the .RMD file extension be added to the file so it may be displayed within the "Command Files" menu within the Database Explorer's Navigator. Now, in R:BASE 11 for Windows, the new standard is .RMD as more file servers and mail servers flag the .CMD file extension as a possible virus. If you are using a custom file

Copyright © 1982-2024 R:BASE Technologies, Inc.

extension, you can add the extension to the "File Mask:" field under the "Command Files" option and they will then appear in this window.

In order to use your command files in R:BASE 11, you will need to review and update the command syntax used in the files. Based on the R:BASE version you are upgrading from, there are additional steps needed to upgrade your command syntax. The older your command files are, then the more steps will be needed. And, if you are upgrading from a DOS version, there are additional changes required as well.

With command files as the primary storage format for the code, it is recommended that you open the R:BASE Editor Help file and review the many features available to add more productivity to the conversion process.

Also, in R:BASE 11, there are several different ways to run the menus contained within the files. Please review the several Application Formats available.

1.9.4.1.3 Updating the Command Syntax

At this point, it is necessary to review and update the command syntax so it can be used within R:BASE 11. If the syntax is used to RUN another file, then that file must also be checked and updated, if needed. Based upon the R:BASE version you are upgrading from, there are additional steps needed to update your application code. The older the application is, then the more steps will be needed. And, if you are upgrading from a DOS version, there are additional changes required as well.

You may encounter reserved words (PROJECT, TIME), obsolete commands (SET POINTER), and/or commands whose syntax has been enhanced for your benefit (PRINT, SELECT, CHOOSE). The time invested to perform these changes depends on the size of your command files.

There are two important tools included with R:BASE 11 that will make your command syntax changes more pleasant that you think.

The new R:BASE Editor & R:Style!

The R:BASE Editor is an editor specifically designed for creation and editing of R:BASE program and application files, command files, recorded scripts, and text files. It offers these special R:BASE features:

- You can specify settings such as the font, tab stops, and text wrapping. You can also search and replace text, and print files.
- Any number of program files can be open simultaneously.
- Command files are shown with syntax highlighting with the option of displaying each syntax element in a different color text.
- Safe tabs -- the Tab key inserts spaces.
- Block commands include block indent, block outdent, block comment out, and block comment in.
- Seamless integration with the R:Style, R:BASE code styling and checker.
- And more...

What is R:Style?

R:Style provides structural error checking, and spelling assistance to help debug your code prior to running it. R:Style will tell you where in the file to look, and what error to look for! R:Style is not a full syntax checker, but it is an excellent structure checker, with some syntax checking thrown in as a bonus. R:Style will find:

- Checks illegal or ambiguous commands
- Checks missing, unmatched, or incorrect quotes
- Checks unmatched parenthesis
- Finds missing RETURNs & ENDs, adds them
- Finds missing or out of place structures, IF, THEN, ELSE, ENDIF, SWITCH, ENDSW, WHILE, THEN, ENDWHILE, BREAK
- Duplicate block names in APP files
- Checks illegal use of command words in code or block names
- Checks ill advised RETURN from within WHILE or SWITCH
- Checks illegal dotted variables
- Checks dash comments extending beyond column 79 (breaks codelock)
- Support for finding misspelled or illegal words

Copyright © 1982-2024 R:BASE Technologies, Inc.

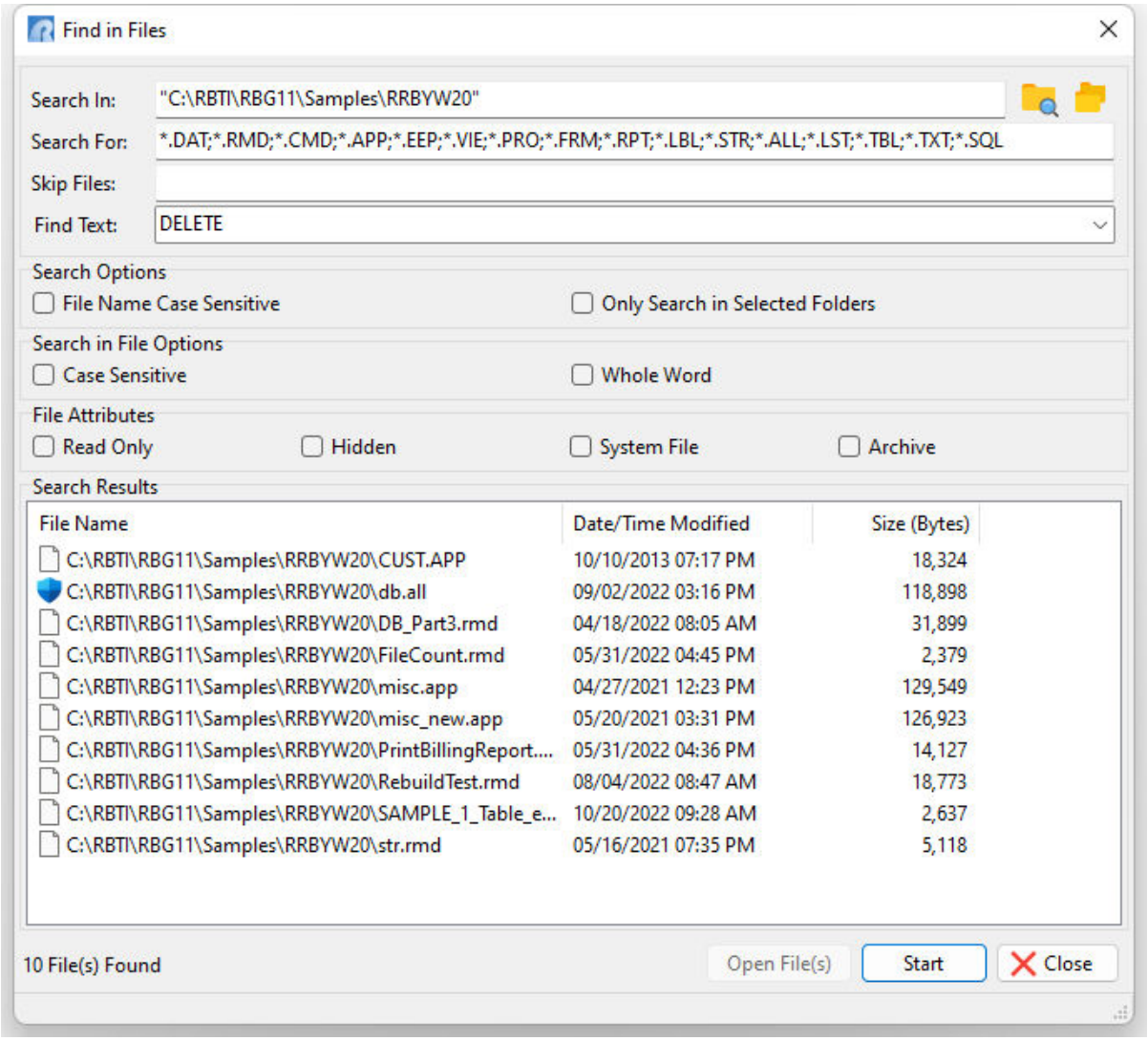
- Integration of RStyle* .PRE, .EXP, .CAS, .VAR, .NEW keyword formatting files into a built-in control panel
- Misterminated lines (= instead of +)
- And more...

1.9.4.1.3.1 Table and Column Name Changes

During the initial Preparation stages of the conversion process you were instructed to review change your table and column names which match any invalid names and reserved words, and to record these changes on paper.

If you made any table and/or column name changes, the first step in your command syntax updates is to make these necessary name changes in the application/command files.

For command files, one time-saving suggestion for this task is to use the R:BASE built-in "Find Files" utility. To launch this command file utility, choose "Utilities" > "Find Files" from the main Menu bar. This interface will allow you to search your command files for obsolete commands and other words, so you are not forced to manually search the individual files.

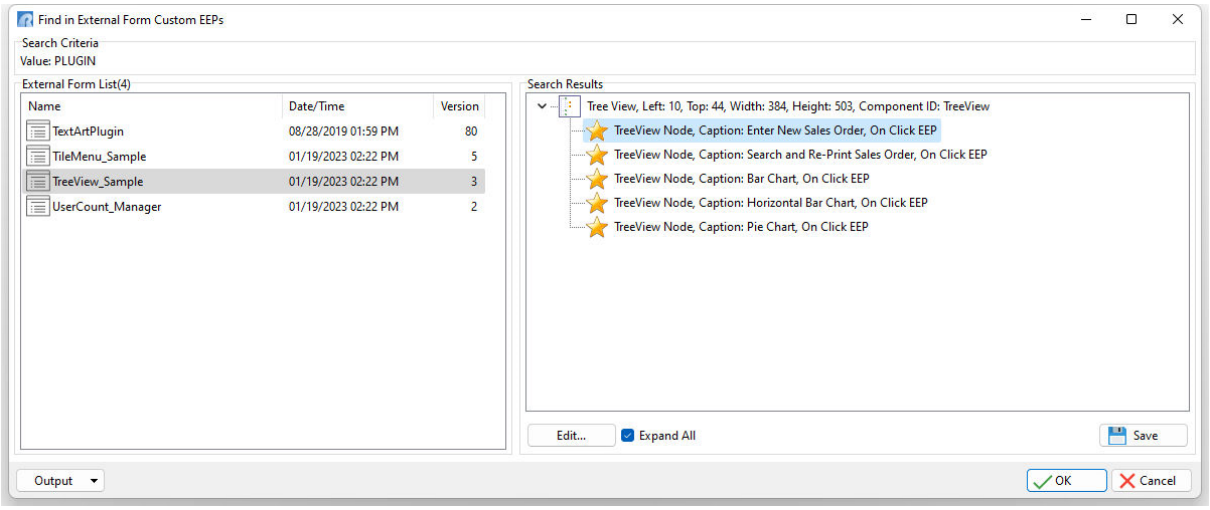


Copyright © 1982-2024 R:BASE Technologies, Inc.

Keep in mind that whether you are using the R:BASE Editor or another text editor to make these command syntax updates, it is wise to take advantage of all of the utilities available in the text editor. In this case where you are replacing keywords, be sure to use any built-in word Search and Replace utilities to make this task easier.

For Custom Form Actions, a built-in search utility is available to search all of the code within the Actions. From within the External Form Designer, select "Edit" > "Find in Custom EEPs" from the main Menu Bar to search for text within Custom EEPs for the external form. The key combination [Ctrl]+[F] can also be used used to launch the "Find in Custom EEPs" dialog.

After the search is complete, a list of external form objects and custom form actions that match the search will be displayed. The external form object(s) will contain the exact location where the Custom EEP is in use for the object. The command syntax can be launched in the R:BASE Editor by double clicking on it, or by selecting the "Edit..." button. The search results can be saved to several output formats. The "Expand All" check box allows the list of EEPs in the search results to expand for full viewing.



1.9.4.1.3.2 Obsolete Commands

The following table contains a list of commands that are considered to be obsolete. Most of these commands have been made obsolete by their SQL Compliant equivalent, and there is no guarantee that these commands will be supported in future versions of R:BASE. Any code which relies on any of these commands should be considered dangerous as their use may have unintended and unpredictable results.

While most items have a direct replacement there are some items that are listed as having no direct replacement. However it may be possible to replace those commands with other commands that emulate that functionality. For more information on the "Replacement" commands, refer to the R:BASE Help "Command Index", or press [F1] from any location in R:BASE.

Copyright © 1982-2024 R:BASE Technologies, Inc.

One suggestion for task this task is to use the R:BASE built-in "Find in Files" utility. To launch this command file utility, choose "Utilities" > "Find in Files" from the main Menu bar. This interface will allow you to search your command files for obsolete commands and other words, so you are not forced to manually search the individual files.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Obsolete Command	Replacement	Notes
#FORM_COLUMNNAME	RBTI_FORM_COLNAME	See also RBTI System Variables
#FORM_FORMNAME	RBTI_FORM_FORMNAME	See also RBTI System Variables
#FORM_TABLENAME	RBTI_FORM_TBLNAME	See also RBTI System Variables
AVE()	LAVG	
BUILD KEY	CREATE INDEX	
CHANGE	UPDATE	
CHANGE COLUMN	ALTER TABLE	
CLOSE	DISCONNECT	
COLUMNS	CREATE TABLE	
(CVAL('FORM_ALIAS')	RBTI_FORM_ALIAS	See also RBTI System Variables
(CVAL('FORM_CAPTION')	GETPROPERTY RBASE_FORM CAPTION vCaption	
(CVAL('FORM_DATA')	RBTI_FORM_COLVALUE	See also RBTI System Variables
(CVAL('FORM_DATA_TYPE')	RBTI_FORM_DATATYPE	See also RBTI System Variables
(CVAL('FORM_FIELD_NAME')	RBTI_FORM_COLNAME	See also RBTI System Variables
(CVAL('FORM_NAME')	RBTI_FORM_FORMNAME	See also RBTI System Variables
(CVAL('FORM_TABLE')	RBTI_FORM_TABLENAME	See also RBTI System Variables
DEFINE	CREATE SCHEMA	
DELETE KEY	DROP INDEX	
DELETE FROM #n	DELETE FROM TableName WHERE CURRENT OF CURSOR	See also DECLARE CURSOR
DISPLAY	EDIT USING	Used with Variable Forms.
DRAW	NONE	Used with Variable Forms. No direct replacement.
ECHO	SET ECHO ON	
EDIT VAR ...	NONE	Used with Variable Forms. No direct replacement.
ENTER VAR ...	NONE	Used with Variable Forms. No direct replacement.
ESC	[ESC]	Used in IF and WHERE clauses to compare to the Esc key.
Value1 EQ Value2	Value1 = Value2	Value1 equals Value2
EXISTS	IS NOT NULL	
EXPAND	ALTER TABLE	
EXPRESS	RBAPP	
FAILS	IS NULL	

Copyright © 1982-2024 R:BASE Technologies, Inc.

FILLIN	DIALOG	In windows consider using DIALOG, ENTER or EDIT.
Value1 GE Value2	Value1 >= Value2	Value1 greater than or equal to Value2.
Value1 GT Value2	Value1 > Value2	Value1 greater than Value2.
IN #n	WHERE CURRENT OF CURSOR	Used with UPDATE and SET VAR commands.
Value1 LE Value2	Value1 <= Value2	Value1 less than or equal to Value2.
Value1 LT Value2	Value1 < Value2	Value1 less than or equal to Value2.
MAX()	LMAX()	
MICRORIM_RETURN	STP_RETURN	
MICRORIM_WALLPAPER	NONE	This variable was only used with Forms and Reports created for R:WEB.
MIN()	LMIN()	
Value1 NE Value2	Value1 <> Value2	Value1 does not equal Value2.
NEXT	FETCH	See also DECLARE CURSOR
NOECHO	SET ECHO OFF	
OPEN	CONNECT	
OWNER	CREATE SCHEMA	
PASSWORDS	GRANT	
PLAYBACK	NONE	No Replacement.
POPUP/PULLDOWN Menus	NONE	Use #LIST option of CHOOSE command. Use LISTOF command option to create list of items.
PROMPTS		Now DOS Only.
RECORD	NONE	
REDEFINE	ALTER TABLE	
REMOVE	DROP	
RESET	NONE	No Replacement.
#RETURN	NONE	Used with Variable Forms. No Direct Replacement.
SCREEN RESTORE	RECALC VARIABLES	Now DOS Only.
SELECT FROM #n	SELECT FROM TableName	See also DECLARE CURSOR WHERE CURRENT OF cursor
SET CGA	NONE	No Replacement.
SET EXTENDED	NONE	R:BASE always operates in Extended Mode now.
SET INTENSITY	NONE	Use "Selected Row" color option in Scrolling Region Properties.
SET OLDLINE	NONE	Use the Page Style option in Reports.
SET POINTER	DECLARE CURSOR	

Copyright © 1982-2024 R:BASE Technologies, Inc.

SET RBGSIZE		No replacement necessary due to imbedded form properties. Command is ignored.
SET USERAPP	NONE	No replacement.
SNAP	NONE	No replacement.
SORTED BY	ORDER BY	
USER	CONNECT IDENTIFIED BY ...	
VIEW	CREATE VIEW	
WRITE		In Windows consider using DIALOG, PAUSE, ENTER or EDIT.
ZIP RBEDIT	RBEDIT	RBEDIT is no longer a separate executable.
ZIP RBSYNC	RBSYNC	RBSYNC is no longer a separate executable.

1.9.4.1.3.3 DOS to Windows Conversion

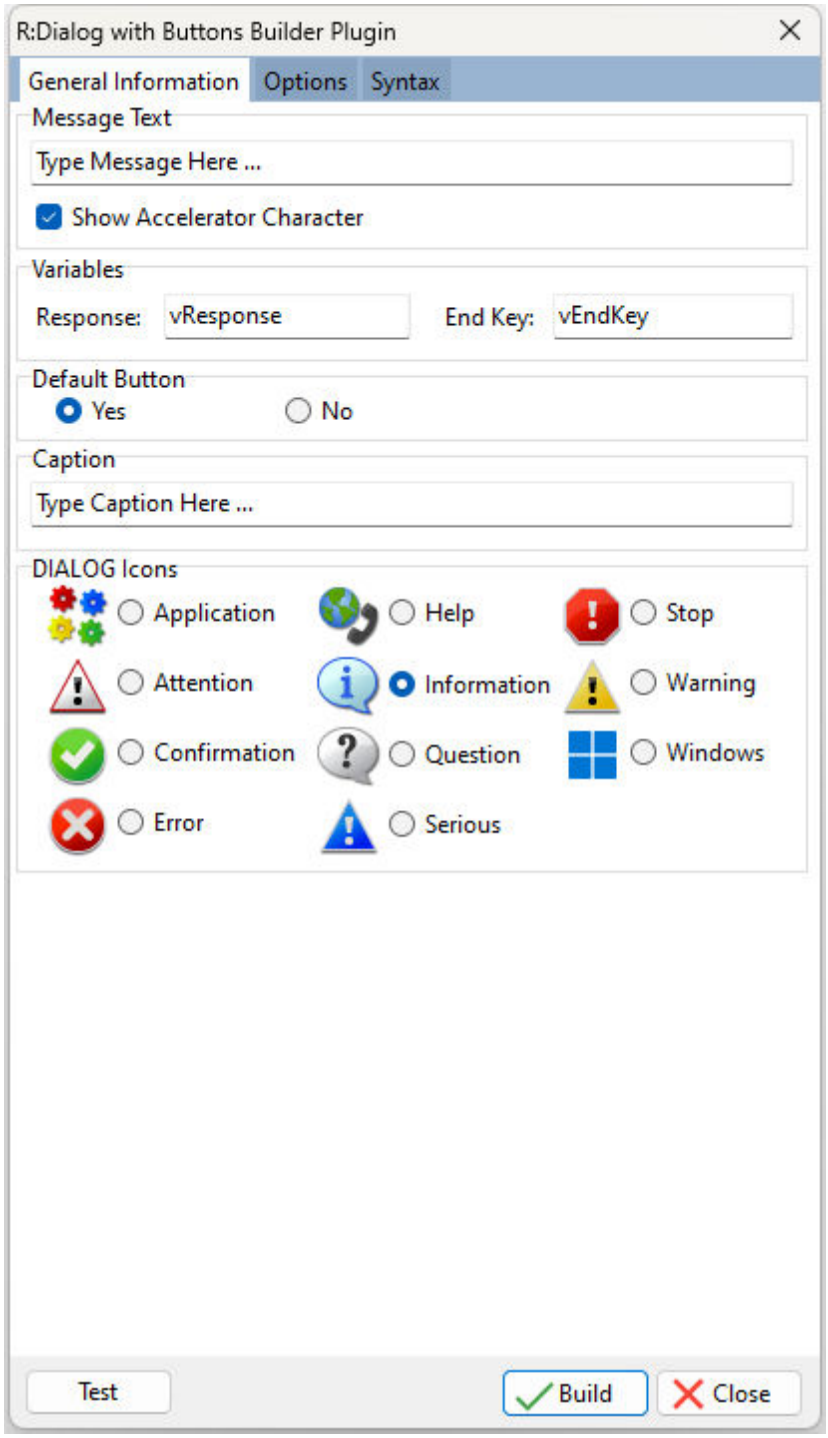
When converting from DOS to Windows, there are command syntax changes that may or may not require your attention.

Instances where you will positively update the command syntax are any locations where DOS message commands (WRITE) or information prompts (FILLIN) which should be replaced by their Windows counterpart (PAUSE and DIALOG).

These commands send simple messages or prompts for data entry to the end user. Your time involved in this step depends on how much user interaction is in the code.

To assist you with these replacements, R:BASE 11 for Windows includes several syntax builders for the DIALOG, PAUSE, and CHOOSE commands. They can be accessed from the Menu Bar under "Utilities" > "Plugins".

Copyright © 1982-2024 R:BASE Technologies, Inc.



For **WRITE**, you may have this;

```
WRITE 'THE PROCESS IS COMPLETE!' AT 10 05
```

and will replace it with this:

```
PAUSE 2 USING 'THE PROCESS IS COMPLETE!' CAPTION ' ' ICON INFO
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

For **FILLIN**, you may have this;

```
FILLIN vdte USING 'ENTER DEADLINE DATE: ' AT 12 10
```

and will replace it with this:

```
DIALOG 'ENTER DEADLINE DATE:' vdte venke 1 CAPTION 'Question'
ICON QUESTION
```

The PAUSE and DIALOG samples provided above do not include the many options that are available for each command to professionally display a message on the screen. Please review the "All About The Dialog Command" and "All About The Pause Command" PDF documents for the complete syntax available for each command. The PDF documents are located within the default program directory of R:BASE 11 (C:\RBTI\RBG11).

1.9.4.1.3.4 Enhanced Commands

There will be commands whose syntax has been enhanced for your benefit. Most of these commands will not require mandatory changes other than for an enhanced display (CHOOSE, DIALOG, PAUSE), but the PRINT command is an example of a command that has been enhanced 20 times over, and will require you to alter your syntax.

You may have this:

```
OUT PRINTER
PRINT bulletin SORTED BY poscl
OUT screen
```

and will replace it with this:

```
PRINT bulletin ORDER BY poscl ASC OPTION PRINTER
```

or even this:

```
PRINT bulletin ORDER BY poscl ASC OPTION PRINTER +
|SHOW_CANCEL_DIALOG OFF |TRAY 2 |COLLATION ON +
|COPIES 2 |ORIENTATION LANDSCAPE |PRINTER_NAME HPLaser1345
```

1.9.4.1.3.5 Unsupported Command Parameters

The application file may have $MENU blocks where the CHOOSE command is used to display a menu. The CHOOSE command "IN MENU.APX" is no longer supported for "PULLDOWN" menu blocks.

If the approach of the application file conversion is to run the .APP file as a command file, then the following method can be used to move the "menu list" into the CHOOSE command with the #LIST parameter.

Instead of the following:

```
CHOOSE PICK1 FROM MENU0000 IN Menu.APX
--
-- rest of file
--
$MENU
MENU0000
PULLDOWN | |
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
|Entry|
|Product Quality Plan|
|Inquiry|
|Reports|
|Maintenance|
ENDC
```

You would use:

```
CHOOSE PICK1 FROM #LIST 'Entry,Product Quality
Plan,Inquiry,Reports,Maintenance' +
TITLE 'SELECT OPTION...' CAPTION 'Main Menu Options' LINES 5 +
OPTION List_Font_Color NAVY|List_Back_Color WHITE|List_Font_Size 12 +
|Title_Font_Color NAVY|Title_Back_Color LIGHT GRAY +
|Window_Back_Color LIGHT GRAY|Title_Font_Size 14
```

The above will display a choose pop-up window to select the desired option.

Although this will allow the application file to work, we recommend using one of the current application formats to display a Windows driven menu system.

1.9.4.1.4 New Application Formats

When building R:BASE databases and applications, R:BASE users have several options as to how they want to store their "application".

1. **Forms** - an application can be completely Form driven by storing all of the code in form EEPs.
2. **Application Builder** - using the Application Builder interface, an application can be stored in a .RBA file, separate from the database files. This offers additional security over using just command files.
3. **Command Files** - using only code (e.g. CHOOSE, DIALOG, PAUSE, etc), an application can be entirely written in one or many RMD, APP, CMD, and DAT files.
4. **External Form Files** - an application can be driven using External Forms, by storing all of the code in form EEPs. Like an R:BASE Application file (.RBA), External Form Files are stored outside of the database.
5. **All, or some, of the above** - an application can consist of all of a few of the above methods to store R:BASE menu and command files.

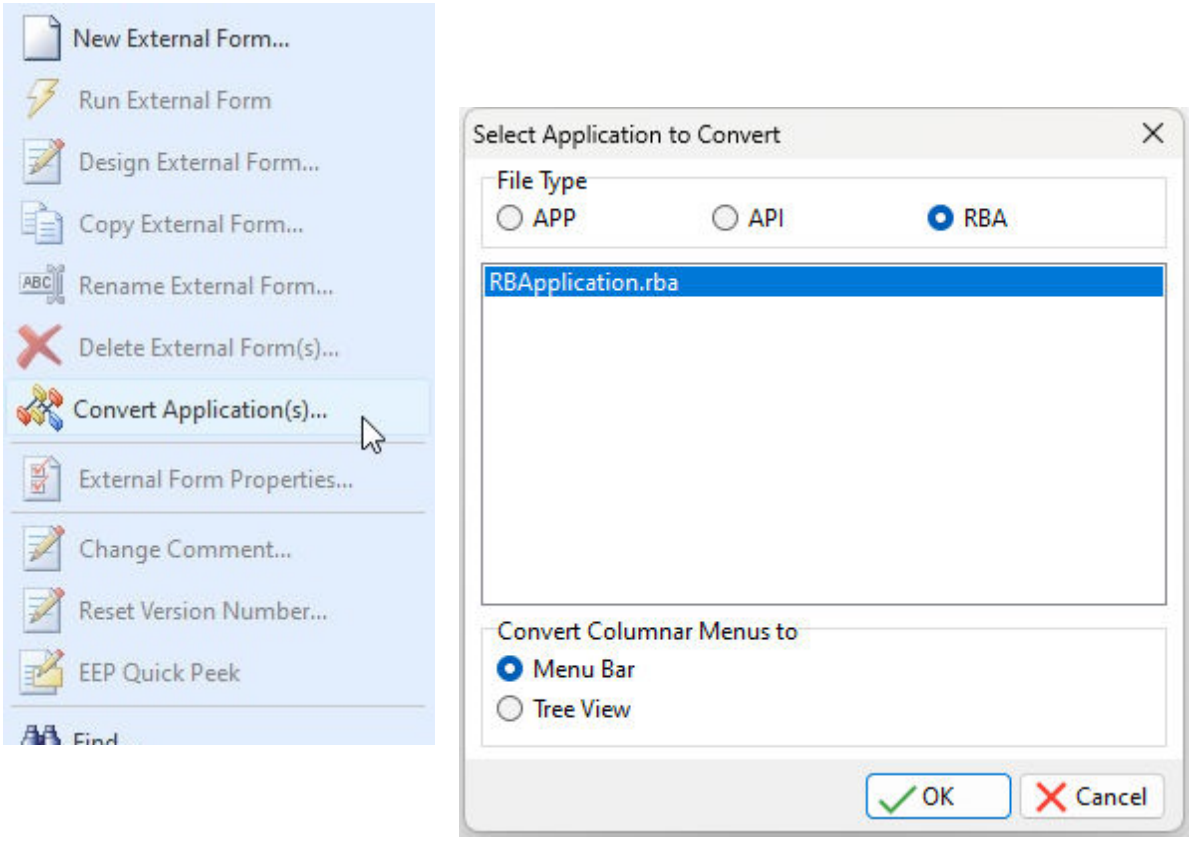
1.9.4.2 R:BASE 7.x/Turbo V-8 Applications (.RBA)

Any .RBA Application files, Form-Driven Applications, External Form Files, or command files running in R:BASE 7.x or Turbo V-8 will run in R:BASE 11 with little or no changes. The R:BASE 7.x versions include 7.0, 7.1, 7.5 and 7.6.

It is recommended that you review your application menus and actions to ensure the system is 100% operational.

Also, within the External Form File portion of the Database Explorer, the new "Convert Application(s)..." option is available to convert R:BASE application files (.RBA) to an External Form File.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Selecting this option will create an external form that matches the same properties used within the RBA application file. For the "Menu Bar" layout, the "Main Menu" section is converted to a form "Menu Bar", the "Actions" are converted to "Custom Form Actions", and the "Toolbar" is converted to a toolbar in the "Toolbar Designer" when the external form file is created. A "Tree View" control is also available as a menu display method, where menu options are loaded into a hierarchical list with "Actions" placed and referenced into Custom EEPs and Custom Form Actions.

Plugins
When upgrading from 7.6/V-8 to R:BASE 11, you will also need to upgrade any Plugins, if used. The older .RBL Plugin files cannot be used in R:BASE 11 as a different plugin type and file extension (.RBM) is used.

The free Plugins that are shipped with R:BASE are contained within the R:BASE 11 executable. Initially, you will not see any Plugins in the R:BASE program folder. The Plugins contained within the R:BASE executable include:

- LoadDirectoryName
- LoadFileName
- LoadFileNamePlus
- RCalculator

When you run the PLUGIN command with your R:BASE program, the window will display appropriately.

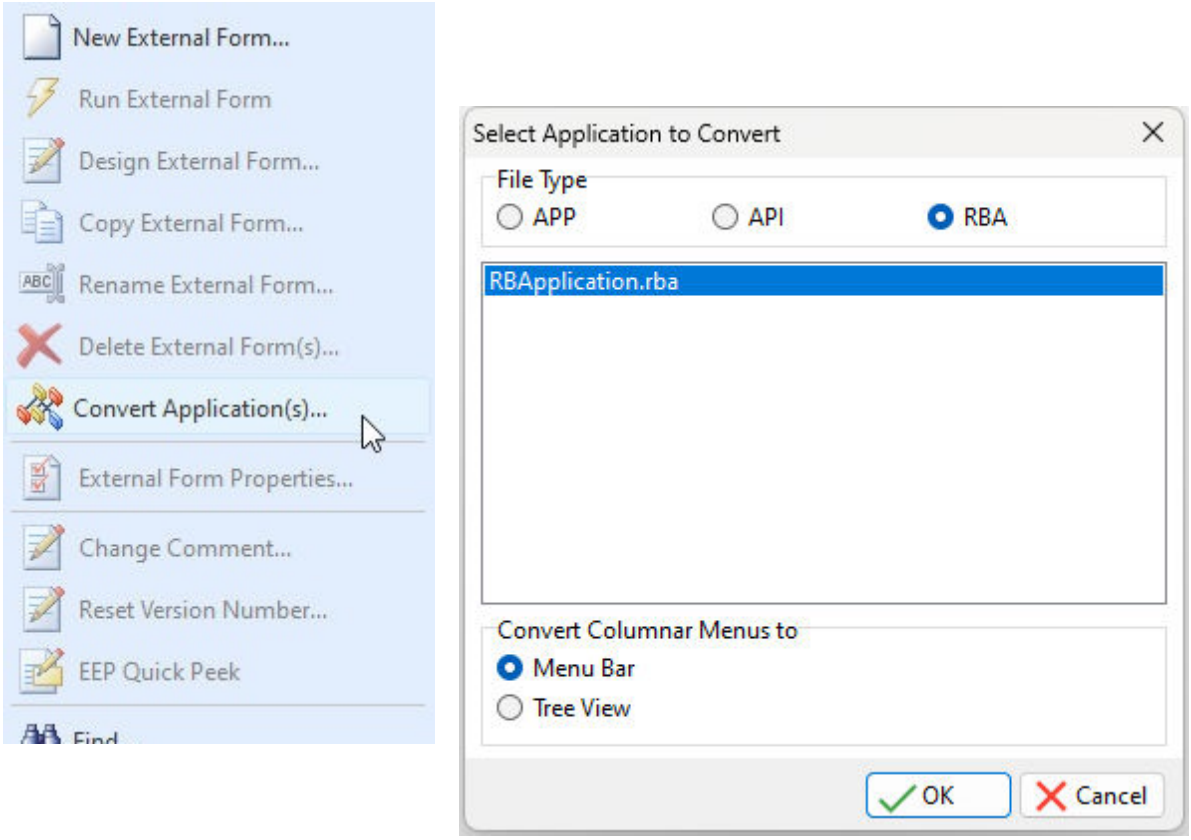
Please refer to the PLUGIN section within the Command Index to verify the supported syntax.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.9.4.3 R:BASE eXtreme 9.x Applications

Any .RBA Application files, Form-Driven Applications, External Form Files, or command files running in R:BASE eXtreme will run in R:BASE 11 with no changes required on your behalf. It is recommended that you review your application menus and actions to ensure the system is 100% operational.

Within the External Form File portion of the Database Explorer, the new "Convert Application(s)..." option is available to convert R:BASE application files (.RBA) to an External Form File.



Selecting this option will create an external form that matches the same properties used within the RBA application file. For the "Menu Bar" layout, the "Main Menu" section is converted to a form "Menu Bar", the "Actions" are converted to "Custom Form Actions", and the "Toolbar" is converted to a toolbar in the "Toolbar Designer" when the external form file is created. A "Tree View" control is also available as a menu display method, where menu options are loaded into a hierarchical list with "Actions" placed and referenced into Custom EEPs and Custom Form Actions.

Plugins
When upgrading from R:BASE eXtreme 9.x to R:BASE 11, you will also need to upgrade any Plugins, if used. The 9.x specific Plugin files cannot be used in R:BASE 11.

1.9.4.4 PAGELOCK Setting

R:BASE 10.5 and higher versions include a new (actually separate) PAGELOCK operating condition, which specifies how R:BASE locks data in a multi-user environment, based on the position of the row in the data file?

Previously, the R:BASE operating condition was controlled by QUALCOLS as a dual-functioning setting, specifying unique columns for ODBC tables and for page locking. Now page locking (PAGELOCK) is separate from ODBC unique columns (QUALCOLS).

Copyright © 1982-2024 R:BASE Technologies, Inc.

What does this mean for R:BASE users?

The application code must be updated to add PAGELOCK and set the condition to ON/OFF appropriately where QUALCOLS was used for page locking. Examples below demonstrate the setting could also be used dynamically in case-by-case instances to increase multi-user performance.

To summarize the code changes, if you used SET QUALCOLS 10 for faster page locking, add SET PAGELOCK OFF to your code. If you used SET QUALCOLS 2 to enforce a page lock for a faster UPDATE/DELETE process, add SET PAGELOCK ON in your code.

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

The settings for PAGELOCK are:

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

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

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

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

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

Notes:

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

Example:

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

1.9.5 ODBC Compliance

If you are planning on using ODBC with R:BASE, you must make sure that your character settings are set correctly by performing the following:

1. Go to the R> Prompt
2. Connect to your database
3. Type: SHOW CHAR
4. Compare your values to the chart below

Copyright © 1982-2024 R:BASE Technologies, Inc.

Character	Set To	Description
MANY	%	Percent Sign
SINGLE	_	Underscore
QUOTES	'	Single Quote

5. If they do not match, change them by typing SET TYPE=CHAR
(where TYPE is from the "Character" column and CHAR from the "Set To" column)

For example, SET MANY=%

Continue below to ensure your database files are **Y2K** compliant.

1.9.6 Y2K Compliance

For more information on Y2K Compliance please visit our Y2K Page at
http://www.rbase.com/support/y2k/

To check the Y2K compliance of your database do the following.

1. Go to an R> Prompt
2. Connect to your Database
3. Type SHOW DATE

Most likely the settings will be as follows... (This is the default for upgraded databases).

```
DATE FORMAT MM/DD/YY
DATE SEQUENCE MMDDYY
DATE CENTURY 19
DATE YEAR 0
```

If so, then your database is NOT Y2K compliant. To activate the Y2K compliancy features of your database we recommend the following:

Set your Format to 4 Y's
When they are set that way then 1/1/1999 will be seen as a valid year and 1/1/99 will also be a valid year. If you are set to 2 Y's then only 1/1/99 will be a valid year.

Set the Date Year to a Convenient Year
The Date Century and Date Year determine how 2 digit years such as 1/1/99 are handled. To determine your settings mentally combine the century and the year (in this case 19 and 0 make 1900) and then add 99 (in this case 1999) all two digit years will be interpreted as being between these dates. Setting Date Year to 80 will result in 19 + 80 giving 1980 and then adding 99 making 2079. Thus all two digit dates will be between 1980 and 2079. For example a date entered as 1/1/60 would be seen as 2060.

Older versions set to four Y's would see 2 digit dates as being complete four digit years. For example, 1/1/99 would be January First in the year 99. This is known as First Century data. For this reason if your database is already set to four Y's on the format we recommend using the following command to check for invalid dates:

```
TALLY datecol FROM tablename WHERE datecol < 1/1/1950
```

This should build a list of all dates (and how often they occur) that are possibly incorrectly entered and that you can then deal with. Be careful not to go overboard. For example: if you have your family tree stored in an R:BASE Database then be careful that you leave Great Grandmother's birthday in the 1800's where it is supposed to be even though the Tally command above would flag that as an "invalid" date.

Copyright © 1982-2024 R:BASE Technologies, Inc.

To resolve data issues where date values are listed as 1900, 1901, etc., where the expected value is 2000, 2001, etc., please continue.

1. Disconnect and make a backup copy of your database.
2. Start R:BASE and connect to the database.
3. At the R> prompt, type:

```
  UPDATE TableName SET DateColumnName = (ADDYR(DateColumnName,100)) +
  WHERE (IYR(DateColumnName)) IN (1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907,
1908, 1909, 1910)
```

Where:

"TableName" is the name of table
"DateColumnName" is the name of date column in that table

1.9.7 Database Integrity Routine

Follow the steps below and to validate the integrity of your database.

1. Make a backup copy of the database.

2. Copy the entire database to your "local drive" in a separate folder with at least twice as much available free disk space.

3. Start R:BASE using the latest version/update and switch the current folder to the appropriate database folder on your local drive.

4. At the R>, connect to the database:

```
DISCONNECT
SET MESSAGES ON
SET ERROR MESSAGES ON
SET MULTI OFF

CONNECT dbname
or connect to the database with OWNER password, if any
CONNECT dbname IDENTIFIED BY owner password
```

5. Check the connected database CHARacter settings, especially the IDQUOTES. At the R> prompt:

```
CLS
SHOW CHAR
```

Notice the setting for IDQUOTES (the last item on list).

If this setting is blank/null, make sure to set the IDQUOTES settings to ` (that is a single reversed quote, on the same key as the ~ tilde). At the R>, type:

```
SET IDQUOTES=`
```

6. Now create the unload file with NULL set to -0-. At the R> prompt:

```
CLS
SET NULL -0-
OUTPUT newdb.ALL
UNLOAD ALL
OUTPUT SCREEN
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

This step will create two files (newdb.ALL and newdb.LOB)

7. DISCONNECT, and then rename this database to some other name. At the R> prompt:

```
DISCONNECT

RENAME dbname.RX? dbnameBK.RX?
or for RB1 through RB4 files, use:
RENAME dbname.RB? dbnameBK.RB?
```

8. Now, let's rebuild the fresh database and see if it passes the integrity check. At the R> prompt:

```
CLS
RUN newdb.ALL
```

Watch the activity and all messages on the screen. Don't fall asleep. You might miss an important warning. Completion time may vary based on size of the database (number of tables/records/indexes)

If there were no warnings or error messages, you've got a fresh/healthy database.

Errors?

1. If there are any warning or -ERROR- message(s), take them seriously. In that case, disconnect and delete the bad database (built using RUN newdb.ALL). At the R> prompt:

```
DISCONNECT

DELETE dbname.RX?
or for RB1 through RB4 files, use:
DELETE dbname.RB?

DELETE newdb.ALL
DELETE newdb.LOB
```

2. Rename and connect to the previously saved database (Step 7).

```
RENAME dbnameBK.RX? dbname.RX?
or for RB1 through RB4 files, use:
RENAME dbname.RB? dbnameBK.RB?

CONNECT dbname
```

3. CORRECT all -ERROR-s accordingly and then repeat Step 6.

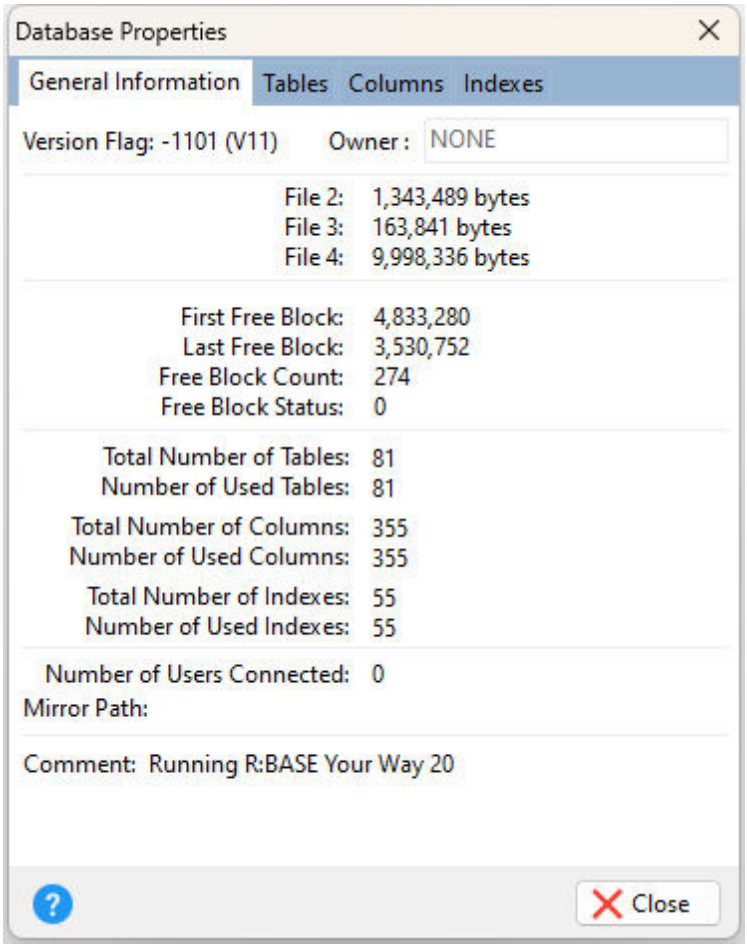
4. Do not quit or give up until you see a completely fresh database without any warnings or errors.

If there is a considerable amount of errors with both rebuilding the structure and/or data, please refer to the "Database Repair Routine" within the Database Maintenance PDF document in the R:BASE program directory.

1.10 Database Properties

The Database Properties menu option within the Databases area of the Database Explorer open a dialog that displays important information for the selected database. When selecting the option a fast connection is made to the database files to retrieve the current information. That connection may temporarily be displayed in the information. The dialog will display details regarding general information, tables, columns, and indexes.

Copyright © 1982-2024 R:BASE Technologies, Inc.



General Information:

- Version Flag
- Database Owner (only if NONE)
- File 2 Size
- File 3 Size
- File 4 Size
- First Free Block
- Last Free Block
- Free Block Count
- Free Block Status
- Total Number of Tables
- Number of Used Tables
- Total Number of Columns
- Number of Used Columns
- Total Number of Indexes
- Number of Used Indexes
- Number of Users Connected
- Mirror Path
- Comment

Tables Information:

- System Tables
- Tables
- Views
- Table Type (Table, System View, View)
- Column Count
- Row Count
- Number of users currently in Row Lock mode

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Number of Cursors open for the table
- Number of users in Transaction mode

Column Information:
- Datatype
- Column Length

Index Information:
- Key Type: (PK, FK, Index)
- Associated Table
- Associated Column(s)

Database Version Flags

6.x, 7.x, 9.x (32), 10.x	10.x Enterprise, 11.0	Meaning
-601	-1101	Normal mode version flag is set to -1001 when you disconnect a database or exit R:BASE normally.
-602	-1102	Database is in read-only schema mode (STATICDB setting is on).
-603	-1103	Database is in transaction processing (TRANSACT is set on).
-604	-1104	Database is in transaction processing mode and is in need of recovery. Use the RECOVER command to restore the database.
-605	-1105	Database is in transaction processing with compatibility (TRANSACT and COMPATIB is set on).
-606	-1106	Database is in transaction processing mode with compatibility and is in need of recovery. Use the RECOVER command to restore the database.
-611	-1111	Normal mode with FASTFK on.
-612	-1112	Read-only schema (STATICDB on) with FASTFK.
-613	-1113	Database is in transaction processing with FASTFK.
-614	-1114	Database is in transaction processing mode with FASTFK and is in need of recovery. Use the RECOVER command to restore the database.
-615	-1115	Database is in transaction processing with compatibility and fast FK (TRANSACT, COMPATIB, and FASTFK is set on).
-616	-1116	Database is in transaction processing mode with compatibility and fast FK, and is in need of recovery. Use the RECOVER command to restore the database.
-621	-1121	Normal mode with FASTLOCK on.
-622	-1122	STATICDB on with FASTLOCK.
-623	-1123	Database is in transaction processing with FASTLOCK.
-624	-1124	Database is in transaction processing mode with FASTLOCK and is in need of recovery. Use the RECOVER command to restore the database.
-625	-1125	Database is in transaction processing with compatibility and FASTLOCK (TRANSACT, COMPATIB, and FASTLOCK is set on).
-626	-1126	Database is in transaction processing mode with compatibility and FASTLOCK, and is in need of recovery. Use the RECOVER command to restore the database.
-631	-1131	Normal mode with FASTFK and FASTLOCK on.
-632	-1132	STATICDB is on with FASTFK and FASTLOCK.
-633	-1133	Database is in transaction processing with FASTFK and FASTLOCK.

Copyright © 1982-2024 R:BASE Technologies, Inc.

-634	-1134	Database is in transaction processing mode with FASTFK and FASTLOCK and is in need of recovery. Use the RECOVER command to restore the database.
-635	-1135	Database is in transaction processing with compatibility, fast FK, and FASTLOCK (TRANSACT, COMPATIB, FASTFK, and FASTLOCK is set on).
-636	-1136	Database is in transaction processing mode with compatibility, fast FK, and FASTLOCK, and is in need of recovery. Use the RECOVER command to restore the database.

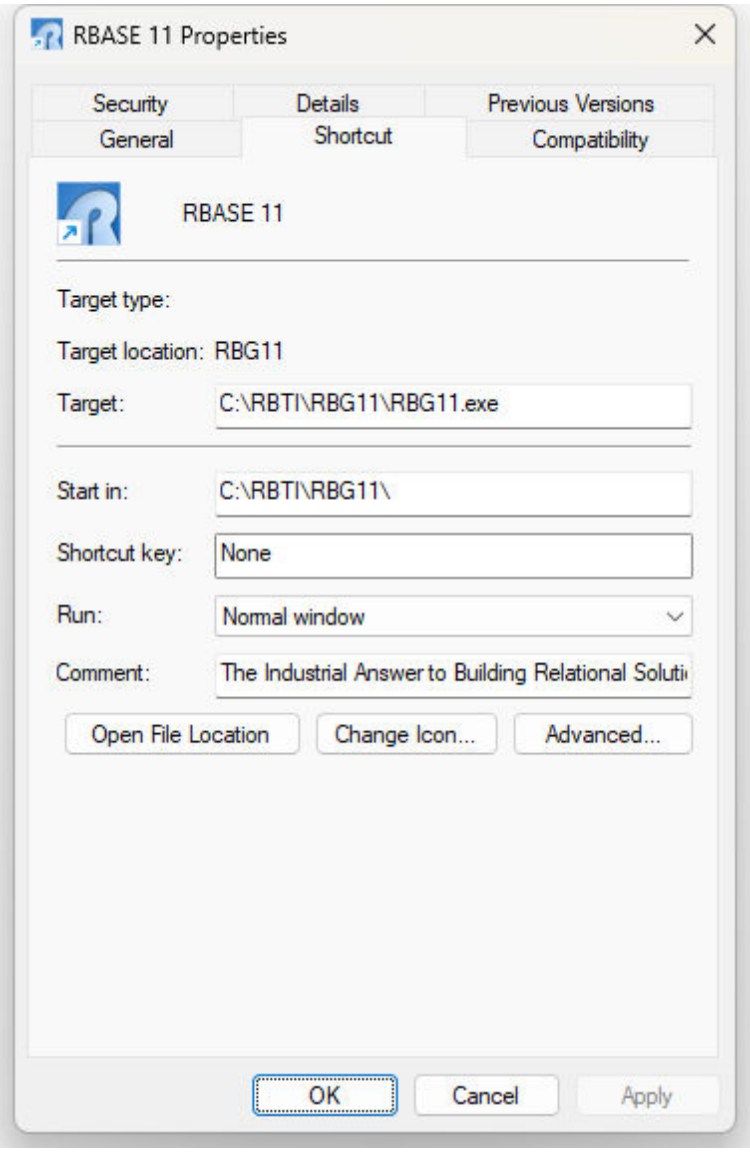
1.11 Desktop Shortcut Properties

R:BASE for Windows supports several startup parameters that users can embed in the desktop shortcut properties, or use as command line switches, to launch a custom R:BASE application, or to launch R:BASE in a different folder. When multiple parameters are specified, each must be separated with a space.

Target:

-A
This option tells R:BASE to look in the specified "Target" folder first for the R:BASE engine and configuration files. This option is beneficial if you are making only one installation of R:BASE on a network server and want local workstations to be able to launch the R:BASE program without the need to install it on the local workstation. Add the "-A" parameter after the executable at the end of the "Target:" field value. The executable and "-A" parameter should be separated with a space.

Copyright © 1982-2024 R:BASE Technologies, Inc.



"-BASE"
Initiates optimized Database Explorer settings for use in active server environments. The settings allow for faster use of R:BASE for Database Administrators who do not have the capabilities to copy an entire database locally just to perform a quick and important change. The values are set on load. This means that when R:BASE is closed the current "minimal" settings will be saved. The following setting changes are performed during load:

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

"-BLOB <filename>"
This option opens the specified file (rich text, image, etc.) in the R:BASE BLOB Editor as the initial screen, when the program is launched. Note that there is a single space between the –BLOB and the file name

Copyright © 1982-2024 R:BASE Technologies, Inc.

parameter, and that the –BLOB <filename> must be enveloped in double-quotes. **Example:** C:\RBTI\RBG11\RBG11.EXE "-BLOB C:\RBTI\RBG11\License.rtf"

-C or **-P**
This option opens the R> Prompt window when the program is launched.

-E
This option opens the R:BASE Editor when the program is launched.

"-E <filename>"
This option opens the specified file in the appropriate file editor when the program is launched. An RFF file is opened in the External Form Designer, an RBA file in the Application Designer, and the rest in the internal R:BASE Editor. Note that there is a single space between the –E and the file name parameter, and that the –E <filename> must be enveloped in double-quotes. **Example:** C:\RBTI\RBG11\RBG11.EXE "-E C:\RBTI\RBG11\ReadMe.txt"

-FS
This option forces the display of splash screen, even if it is set to not be shown in the Startup Options, or if "-S" is used.

-L
This option opens the Database Explorer when the program is launched.

"-O filespec"
Specifies an alternate R:BASE configuration file to use for startup information. Note that there is a single space between the –O and the file name parameter, and that the –O <filespec> should be enveloped in double-quotes. **Example:** C:\RBTI\RBG11\RBG11.EXE "-O C:\Users\William\Documents\RBENGINE11.CFG"

filespec
This option specifies a startup file opens your custom application directly. It can be a command file (.DAT, .RMD, .CMD) or an R:BASE application file (.RBA).

-S
This option opens R:BASE in silent mode, where the splash screen is hidden even if it is set to be shown in the Startup Options.

"-T xxx"
This option adds an additional text label to an R:BASE instance, where xxx is a custom text (e.g. Development) or "RandomID", to better identify the running instance in the Task Manager. This is useful when running multiple instances of R:BASE. A recommended parameter is "-T RandomID" to use the current operating system TickCount value, which returns the number of milliseconds elapsed since the system started. **Example:** C:\RBTI\RBG11\RBG11.EXE "-T RandomID"

-X
This option does not load the Database Explorer, R> Prompt, or R:BASE Editor, even if the modules are set to be shown in the Startup Options. This parameter has a lower priority than –C, -P, -L, and -E. For example, the editor will be shown if both –X and –E is used. This option is useful when R:BASE is used as a file editor, as the program can be loaded with only the R:BASE Editor displayed at startup.

-RBACKUP
This option specifies R:BASE will open as only as R:Backup. When this switch is present, other switches meant for R:BASE are ignored. R:Backup specific parameters include the following:

> **-S**
> Runs R:Backup in silent mode
>
> **-R**
> Runs an R:Backup project
>
> **filespec**
> Specifies the R:Backup file (.rtb) to be loaded or executed

Start in:

Copyright © 1982-2024 R:BASE Technologies, Inc.

By editing the "Start in:" field, you can alter what folder location R:BASE will launch in. You can specify a local folder, a mapped network drive, or a UNC path.

Launching Applications

To launch a custom application on a network folder, edit the "Start in:" field to the database and custom application folder. Then, specify your R:BASE application file or startup file in the "Target:" after the R:BASE executable. Separate the executable and startup file with a space.

Notes:

- The order in which the parameters are listed in the Target field is the order in which the modules will open and appear in the R:BASE program, with the last parameter displayed. For example, using the following Target: C:\RBTI\RBG11\RBG11.EXE -E -L -C , R:BASE will open with the modules in the order of R:BASE Editor, Database Explorer, and R> Prompt, with the R> Prompt window displayed.
- When multiple parameters are specified, each must be separated with a space.

Examples

Example 01:
Launches R:BASE in the RRBYW20 sample database folder, opening the R:BASE Editor, R> Prompt, and Database Explorer modules while displaying the Database Explorer.

```
Target: C:\RBTI\RBG11\RBG11.EXE -E -C -L
Start in: C:\RBTI\RBG11\Samples\RRBYW20
```

Example 02:
Launches R:BASE in the program folder, opening the Database Explorer and R:BASE Editor modules, while displaying R:BASE ReadMe.txt text file.

```
Target: C:\RBTI\RBG11\RBG11.EXE -L "-E C:\RBTI\RBG11\ReadMe.txt"
Start in: C:\RBTI\RBG11
```

Example 03:
Launches R:BASE in the program folder, opening the Database Explorer and R> Prompt modules, while initially displaying R:BASE License.rtf in the BLOB Editor.

```
Target: C:\RBTI\RBG11\RBG11.EXE -L -C "-BLOB C:\RBTI\RBG11\License.rtf"
Start in: C:\RBTI\RBG11
```

Example 04:
Launches locally installed R:BASE in a shared network folder, and launches the CRM.DAT startup file.

```
Target: C:\RBTI\RBG11\RBG11.EXE CRM.DAT
Start in: R:\RBFILES\CRM
```

Example 05:
Launches a network installation of R:BASE in a shared network folder with the -A parameter, and launches the SPECS.DAT startup file.

```
Target: R:\RBTI\RBG11\RBG11.EXE -A SPECS.DAT
Start in: R:\RBFILES\SPECS
```

Example 06:
Launches a network installation of R:BASE in a shared network folder with the -A parameter, uses a custom configuration file for the user, and launches the EST.DAT startup file.

```
Target: R:\RBG11\RBG11.EXE -A "-O R:\RBG11\CFG\JOE\RBENGINE11.CFG" EST.DAT
Start in: R:\RBFILES\EST
```

Example 07:
Launches R:BASE in a development folder on the D:\ drive, with a custom configuration file, and uses a random ID to identify several R:BASE instances within the Windows Task Manager.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Target: C:\RBTI\RBG11\RBG11.EXE "-O C:\Users\Glen\Documents\RBENGINE11.CFG" "-T RandomID"
Start in: D:\RBASEDEV

Example 08:
Launches R:BASE 11 in a database folder on the R:\ drive, and uses text to identify the R:BASE instance within the Windows Task Manager, as per the command file routine. The text "Logistics" will appear next to "R:BASE 11" under the "Applications" tab.

Target: C:\RBTI\RBG11\RBG11.EXE Logistics.rmd "-TLogistics"
Start in: R:\DBFILES

1.12 Displaying Fonts

1.12.1 How R:BASE Displays Fonts

When you display data in R:BASE for DOS, the font is determined by the DOS character set.

When you display data in R:BASE for Windows, the font depends on if you display data in the Data Browser or the R> Prompt:

- The Data Browser uses a default font that was specified when R:BASE was installed. To select a different font, choose "Settings" > "Data Browser" from the main Menu Bar.
- The R> Prompt can display several fonts; OEM, ANSI, and ASCII. The OEM font corresponds to the "Terminal" font, which corresponds to the Windows character set. You can change the R> Prompt settings by choosing "Settings" > "R> Prompt" from the main Menu Bar.

1.12.2 How DOS and Windows Display Fonts

DOS uses a *code page*, or *character set*, to determine which font to use when displaying data. For instance, the United States character set is usually *437*. If a program needs a particular character set, for instance a program that uses French characters, the character set is usually changed by the program itself through a BIOS call.

Windows uses a different system for displaying fonts. However, Windows also uses a code page, or character set, when displaying data in a DOS window. This character set corresponds to the Windows "Terminal" font. You can change the Windows character set when you set up Windows.

1.13 Dotted versus Ampersand Variables

Dotted Variables

1. The variable is used to hold a value.

2. Should be used when R:BASE is to use the "value" contained in the variable.

3. Should be used when R:BASE is to use the variable in a calculation or as a comparison value.

4. A rule of thumb for when to "dot" a variable is to always "dot" the variable when it is on the right side of the operator.

5. Dotting a variable basically turns it into a constant value. R:BASE looks only at the value of the variable when it is dotted. R:BASE doesn't look at the data type, just at the value the variable contains. That's why you can have a TEXT data type variable (result of a CHOOSE command, for example) containing a value that looks like a DATE and use that variable to compare to a DATE data type column or variable

6. In order for R:BASE's expression processing to explicitly see the variable with a TEXT data type, it must be enclosed in parenthesis. Without the parenthesis, the data type is unknown. When R:BASE encounters an unknown data value, it needs to guess what it is. There is a defined sequence that R:BASE goes through in guessing the possible data type. This is the order:

    - INTEGER

Copyright © 1982-2024 R:BASE Technologies, Inc.

- DOUBLE
- CURRENCY
- DATE
- TIME
- DATETIME
- BOOLEAN

7. In expressions, R:BASE checks the data type of a dotted variable. An expression is anything enclosed in parentheses. R:BASE verifies the data type in expressions to make sure the expression is valid. For example, an INTEGER cannot be added to TEXT.

8. Dotted variables are used in Form and Report expressions, just as they are in the SET VAR command. When on the right side of the operator, dot the variable.

9. Dotted variables are commonly used in WHERE clauses and in calculations with other variables.

Example 01:

```
SET VAR vDate DATE = 11/19/2001
SELECT * FROM TransMaster WHERE TransDate <= .vDate
```

The example selects all the records from the TransMaster table where the value in the column TransDate is less than or equal to the value contained in the variable vdate.

Example 02:

```
SET VAR vOrder INTEGER = 20001
SET VAR vAmount CURRENCY = NULL
SET VAR vShip TEXT = NULL
SET VAR vFreight CURRENCY = NULL
SET VAR vTax CURRENCY = NULL
SET VAR vState TEXT = NULL
SELECT NetAmount, ShipMethod, StateAbr INTO +
   vAmount INDIC IvAmount, +
   vShip INDIC IvShip, +
   vState INDIC IvState +
   FROM Orders WHERE OrderNum = .vOrder
IF vShip = 'AIR' THEN
   SET VAR vFreight = $11.00
ELSE
   SET VAR vFreight = $5.00
ENDIF
SET VAR vAmount = (.vAmount + .vFreight)
SELECT TaxRate INTO vTax INDIC IvTax +
   FROM States WHERE StateAbr = .vState
SET VAR vAmount = (.vAmount+(.vAmount*.vTax))
```

On the right side of the equals sign (the operator), the variable is dotted. On the left side of the operator, in the IF and the SET VAR commands, the variable is referenced by its name only, it is not dotted.

Ampersand Variables

1. When a variable contains part of a command, its name is prefaced with an ampersand, "&", and it is called an ampersand variable. The "&" in front of the variable name tells R:BASE that the variable contains part of the command, not a value, and the contents of the variable are used when parsing the command.

2. You cannot dot a variable when it contains part of a command - a table or column name, or an ORDER BY or WHERE clause.

3. Because an ampersand variable is part of a command, it can't be used inside parentheses. Parentheses indicate expressions, expressions are parsed separately from the rest of the

Copyright © 1982-2024 R:BASE Technologies, Inc.

command. You need to include the parentheses as part of the variable value. Sub-selects and IN lists are enclosed in parentheses and you can't use an ampersand variable inside them, you need to include the entire sub-select or IN list, including parentheses, as the variable value.

4. Ampersand variables are most often used to hold table and column names and WHERE and ORDER BY clauses. By using ampersand variables to hold column and table names, you can use the same command to select data from different tables. The CHOOSE command can be used with an ampersand variable to display menus of available tables and columns, as used in Example 01 below.

5. If a variable is used on the left side of the operator, an ampersand variable is to be used.

6. It is important to not confuse the ampersand that prefaces a variable name with the ampersand that is used to concatenate TEXT values.

Example 01:
```
SET VAR vTable TEXT = NULL
SET VAR vColList TEXT = NULL
CLS
CHOOSE vTable FROM #TABLES  TITLE 'Choose Table' +
    CAPTION 'List of Tables' LINES 18 FORMATTED
CLS
CHOOSE vColList FROM #COLUMNS IN &vTable  +
    CHKBOX TITLE 'Choose Column(s)' CAPTION 'List of Columns' +
   LINES 18 FORMATTED
```

The values selected from the CHOOSE commands are placed into variables. The variables might look like this:

```
vTable = Employee TEXT
vColList = Empid,EmpLname,EmpPhone,EmpExt TEXT
```

The variables are then used in any command that uses a table name or column list. Where you see Column names or Table/View name you can substitute an ampersand variable that contains the Column, Table or View name. The variables must be used as ampersand variables to tell R:BASE they contain part of the command.

Example 02:
```
BROWSE &vColList FROM &vTable
```

To prompt for an ORDER BY clause, use the CHKSORT option with the CHOOSE...FROM #COLUMNS.

Example 03:
```
SET VAR vOrderBy TEXT = NULL
CHOOSE vOrderBy FROM #COLUMNS IN &vTable  +
    CHKSORT TITLE 'Choose Column(s)' CAPTION 'Order By' +
    LINES 18 FORMATTED
```

The CHKSORT option prompts for Ascending or Descending order just like the R:BASE sort menus. The variable contains ASC or DESC as well as the column names. It might look like this:

```
vOrderBy = EmpLname ASC,EmpFname ASC TEXT
```

In the command, add the keywords ORDER BY with the ampersand variable containing the columns to order by.

Example 04:
```
BROWSE &vColList FROM &vTable ORDER BY &vOrderBy
```

Example 05:
```
IF &vCodeInput CONTAINS 'New'
  PAUSE 2 USING 'New Parts included.'
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
ELSE
  PAUSE 2 USING 'New Parts are not included.'
ENDIF
```

1.14 Expressions

Expressions calculate values by combining variables, columns, functions, constants, and operators. Expressions define computed columns and assign values to variables. Expressions are mathematical or text formulas used to calculate new values by manipulating existing values.

Expressions combine existing values using operators. Operators used in expressions determine the type of calculations to make. The list of operators that can be used in R:BASE expressions are displayed in the table below:

Operator	Type	Example
+	Unary plus	SET VARIABLE *var1 = +.var1*
-	Unary minus	SET VARIABLE *var1 = -.var1*
**	Exponentiation	SET VARIABLE *var1 = (.var2**2)*
*	Multiplication	SET VARIABLE *var1 = (.var2 * 100)*
/	Division	SET VARIABLE *var1 = (.var2 / 100)*
+	Addition	SET VARIABLE *var1 = (.var2 + 1)*
-	Subtraction	SET VARIABLE *var1 = (.var2 - 1)*
+	Concatenation (w/o space)	SET VARIABLE *var1* = ('light' + 'bulb') Result = lightbulb
&	Concatenation (w/space)	SET VARIABLE *var1* = ('light' & 'bulb') Result = light bulb

You can use any data type in an expression. The binary and character large object data types (BIT, BITNOTE, LONG VARBIT, LONG VARCHAR, VARBIT, AND VARCHAR), although they are valid only in equality operations, are not recommended for use in expressions. The Data Type Results Table listed below displays the data type results when specific data type values are used:

Data Type	Yields
CURRENCY	CURRENCY (However, dividing CURRENCY by CURRENCY yields DOUBLE)
DATE	• DATE – DATE = INTEGER (days) • DATE + INTEGER = DATE • DATE – INTEGER = DATE
DATETIME	DATETIME – DATETIME = DATETIME (smallest specified time unit)
TIME	• TIME – TIME = INTEGER (smallest specified time unit) • TIME + INTEGER = TIME • TIME – INTEGER = TIME
TEXT	Either TEXT or NOTE. A TEXT expression can contain only text strings, text functions, or concatenation operators (+ or &).
NOTE	Either NOTE or TEXT. A NOTE expression can contain only text strings, text functions, or concatenation operators (+ or &).
INTEGER	Depends on data types of the other elements and on order of calculation in the expression. • INTEGER + REAL = REAL • INTEGER + DOUBLE = DOUBLE

See Also

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.14.1 Rules for Defining Expressions

If you use a constant value in an expression and the constant contains one of the separator characters (+, -, *, /, &, **, (,), comma, or space), enclose the constant in quotation marks. For example, if you use a date constant such as 01/15/93 in an expression, enclose the date in quotes: '01/15/93'. Be sure to use the quote character set in the QUOTES setting.

Expressions can contain up to 2000 characters. Each R:BASE command can contain up to 80 expressions, wherever expressions are legal. For example, you can define up to 400 computed columns in a table. An expression can contain a maximum of 100 items (an item can be an operator or an operand).

Expressions can contain one or more R:BASE math functions, each of which counts as an item.

You can use the value of a global variable in an expression by placing a period before the variable name, as in *.variable.*

When an expression is computed, values that exceed the limits of a column or variable are assigned a null value (limits are determined by the defined data type). Any expression that contains a null value produces a null result unless you SET ZERO to ON. When ZERO is on, nulls are treated as zeros. However, when ZERO is on and the expression concatenates text strings, null strings are ignored.

Follow these general rules when defining expressions:

- Begin each variable name with the letter *v* to distinguish variables from column names and constants, for example, *v1* or *vcounter*.
- Do not give a variable the same name as a column.
- Set the data type of a variable before assigning it a value, for example:

```
SET VARIABLE V1 CURRENCY, V2 INTEGER
```

See Also

Examples of Expressions

1.14.2 Examples of Expressions

In the following example, the first command defines the data type of the result variable and assigns an initial value of zero. The second sets the value of *vctr* to the current value of *vctr* plus one. This is the most common method of counting occurrences in a command file.

```
SET VARIABLE vctr INTEGER = 0
SET VARIABLE vctr = (.vctr + 1)
```

In the following example, CREATE TABLE defines the computed column *bonus. Netamount* and *bonuspct* are defined first with CURRENCY and REAL data types, respectively. Bonus is then defined as a computed column, the result of multiplying *netamount* by *bonuspct.*

```
CREATE TABLE salesbonus (netamount CURRENCY, bonuspct +
REAL, bonus = (netamount * bonuspct) CURRENCY)
```

In the following example, the command uses select functions to calculate the total sales amount and total cost. The cost is then subtracted from the price to see the amount of profit.

```
SELECT (SUM(price)-SUM(cost)) from orders
```

See Also

Rules for Defining Expressions

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.15 Entry/Exit Procedures (EEPs)

Entry/Exit Procedures (EEP) are series of commands that can be run within forms, reports, and labels. EEPs provide numerous objectives for versatility in controlling data processing and screen displays.

An EEP can represent a command file that uses the .EEP file extension, or can represent a series of commands embedded with a form, report, or label, which is then referred to as a "Custom EEP". The only difference between an EEP and a Custom EEP is that the Custom EEP is stored in the form, report, or label, whereas a regular EEP must be stored in a command file with the .EEP file extension. An EEP can even be a series of commands that can be executed within a command block of a binary codelocked application file. An EEP can also be a Predefined EEP, which is an EEP option that provides several "predefined" commands when adding buttons to R:BASE Forms.

Custom EEPs were introduced in an effort to decrease the amount of command files stored in a database folder, and to increase portability of a database application. In fact, if you locate all of your EEPs in custom EEPs, you can limit your database folder to just one command file as the startup file. Custom EEPs are now the most common method for storing entry/exit procedures.

EEPs can be executed in the following places:

- Upon entry or exit from a form field
- Upon click, right click, or double click within various form fields
- Upon a key press within some form fields
- Upon the mouse cursor hovering over or leaving a form image
- Upon entry in a row
- Upon exit from a row before updating the row in the database
- Upon exit from a row after updating the row in the database
- Upon changing sections (tables) in the form
- Upon before inserting a row
- Upon after inserting a row
- Before or after a report band
- Before or after a label band
- And more...

EEP Topics:

1.15.1 How to Define an EEP

As an EEP can be defined in different R:BASE modules (forms, reports, labels) using different methods (predefined, Custom EEP, external command file), there are several methods to define an EEP.

1.15.1.1 Form EEPs

Within the Form Designer, EEPs can be defined in various areas based upon how the form or form control is supposed to react to any input or changes made by a user in runtime mode. Once the user makes the specific change, the series of commands within the EEP will execute.

The following Form Designer areas are available to define an EEP:

- **Table Level -** EEPs can be defined to execute before or after changes are made to a table row

Copyright © 1982-2024 R:BASE Technologies, Inc.

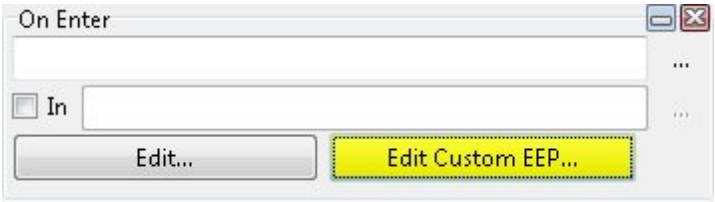
- **Form Control** - EEPs can be defined to execute on nearly every form control where placement of the cursor focus interacts with the control. This includes cursor movement from field to field and mouse pointer events, like clicking on objects.
- **Form** - EEPs can be defined to execute when a form is opened, closed, or resized.

In all of the above areas, there are three different ways to "store" your EEP in R:BASE.

1. **Custom EEP** (stored inside the database files)

   The Custom EEP has become the new standard in the latest R:BASE versions with its ability to store application code within the form. To use a Custom EEP, select the "Edit Custom EEP..." button, which will launch the R:BASE Editor. Enter your code within the editor window, and press the OK button. Then, the "Edit Custom EEP..." button will turn yellow letting you know that code is stored there.

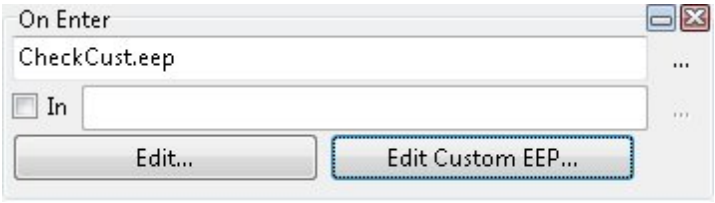
   If security is a concern for the application, consider setting a form "design-time" password. From the Form Designer, choose "Layout" > Passwords" from the Menu Bar.



   Documenting Custom EEPs in Forms
   With several locations to store even the smallest amount of code, the requirement to document where application code lies may become overwhelming. To easily document all Custom EEPs stored within a single form, select "Form" > "Document Custom EEPs" from the Form Designer Menu Bar. Options to generate Custom EEP documentation to the clipboard, printer, text file or PDF file are available.

2. **EEP file** (ASCII command file using the .eep extension stored outside the database files)

   To use an EEP file to store application code, select the "..." button to load a pre-existing EEP file. The "Open" file dialog will look for a file that has a .eep file extension. It is also possible to create a new EEP file directly from this panel by entering a file name within the first field and then selecting the "Edit..." button. Regardless if an new file is being created or an existing file is being edited, the "Edit..." button will launch the R:BASE Editor. After any command syntax changes to your code, select the "Save" button to save the work. The EEP file name will be saved within the field as seen in the image below.

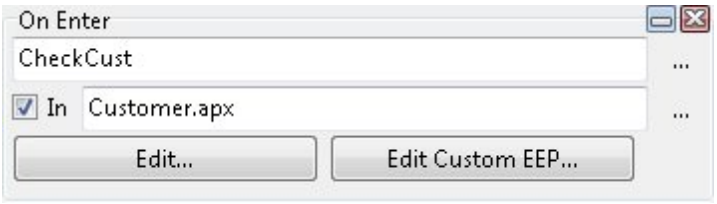


   Storing application code in ASCII files will allow others to open the .eep file with any text editor. If security is a concern for the application, consider storing the application code in a Custom EEP or a procedure file.

3. **Command Block in a CodeLocked Procedure File** (binary file stored outside the database files)

   To use a command block with a codelocked procedure file to store application code, specify the command block name in the first field. Then, select the "In" check box and enter the codelocked procedure file name which contains the command block, into the second field.

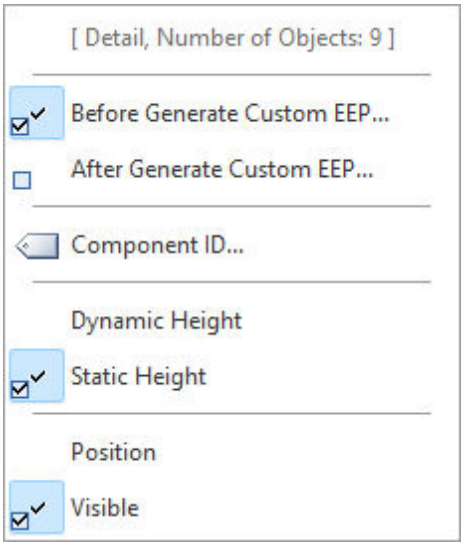
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.15.1.2 Report/Label Custom EEPs

Within the Report/Label Designer, Custom EEPs can be defined within any band that exists upon the report or label. As the R:BASE report/label generator processes the output from top to bottom when printing, an EEP placed on any band can be executed before or after the band.

The following image is the speed menu displayed when the Detail band is right clicked.



Before Generate Custom EEP - the series of commands within the EEP will execute before the band is generated

After Generate Custom EEP - the series of commands within the EEP will execute after the band is generated

1.15.2 EEP Example

The following example of an EEP involves a *family* database comprised of two tables, *family* and *members*, designed to keep track of family members and their ages. The *family* table has columns for family name (*fname*), address (*address*), and city (*city*). The *members* table has columns for a family member's name (*mname*) and age (*age*).

In the *family* form, one field holds the variable *v1*, which keeps a running total of the number of family members. The variable *v1* is set to increment based on the number of members in the members table. A variable *vfname* is set to the *fname* value of the *family* table so that the EEP (FENTRY.EEP) is able to correctly link the rows in the *members* table to the *family* table. The EEP name is placed in the "On Row Entry" box in the Table Settings dialog box in the Forms Designer, and consists of the following expressions:

```
*(fentry.eep)
SELECT COUNT(MName) INTO vCount FROM Members +
    WHERE FName = .vFName
RECALC VARIABLES
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

When the form is run, the FENTRY.EEP is run upon entry in the row. When [F2] is pressed to leave the *fname* field, an EEP (FEXIT.EEP) is run that prompts for a new family member and age to add to the *members* table. The EEP is placed in the "On Exit" EEP field in the "DB Edit" Properties dialog box for the *fname* column of the *family* table, and consists of the following expressions:

```
*(fexit.eep)
SET VARIABLE vKey = (LASTKEY(0))
IF vkey = '[F2]' THEN
    SET VARIABLE vMName TEXT = NULL
    SET VARIABLE vAge TEXT = NULL
    DIALOG 'Enter name of member to add' vMName vEndKey 1 +
        CAPTION 'Criteria...' ICON INFO
    SET VARIABLE DMess = ('Enter the age for' & .vMName)
    DIALOG .DMess vAge vEndKey 1
    SET VARIABLE vInt = (INT(.vAge))
    INSERT INTO Members VALUES (.vFName,.vMName,.vInt)
    SELECT COUNT(MName) INTO vCount FROM Members +
        WHERE FName = .vFName
    RECALC VARIABLES
    RECALC TABLES
ENDIF
```

Upon return to the form from FEXIT.EEP, FENTRY.EEP executes and recalculates the *members* table resulting in the redisplay of the *v1* variable and shows the updated tally of family members, which is accomplished by using the RECALC VARIABLES and TABLES commands.

EEP Topics:

EEP Specific Commands
How to Define an EEP
Field Calculations
EEP Processing Order
Redisplaying Field Values
RECALC Command Options
EEP Restrictions

1.15.3 EEP Restrictions

R:BASE executes a RUN command file when you include a Custom EEP within a form. However, there are restrictions on the command syntax and the types of commands you can use in an entry/exit procedure. In your entry/exit procedures you cannot include the following commands:

COMMIT
INPUT
QUIT
RBDEFINE
UPGRADE
RESTORE
ROLLBACK
RBSYNC
RECOVER
FLUSH
ON CONNECT/PACK
ON DISCONNECT/EXIT
ON ERROR
ABORT
GET
RBAPP

Function Restrictions:

Copyright © 1982-2024 R:BASE Technologies, Inc.

In an EEP that uses the CHKKEY function, you need to replace the CHKKEY function with a PAUSE command. Windows interrupts and takes the key specified by the CHKKEY function before the key can be passed to the CHKKEY function; therefore, the EEP does not receive the information that a key was pressed. In programs run from the R> Prompt, the CHKKEY function runs correctly.

EEP Topics:

EEP Specific Commands
How to Define an EEP
Field Calculations
EEP Processing Order
Redisplaying Field Values
RECALC Command Options
EEP Example

1.15.4 EEP Specific Commands

Nearly all command can be used within an EEP. However, the following commands are used specifically for entry/exit procedures:

DELROW
DUPROW
EXITFORM
FIRST
LAST
NEWROW
NEXTROW
NEXTTAB
PREVROW
PREVTAB
RECALC
RESETROW
RHIDE
RSHOW
SAVEROW
SKIP

EEP Topics:

Field Calculations
How to Define an EEP
EEP Processing Order
Redisplaying Field Values
RECALC Command Options
EEP Restrictions
EEP Example

1.15.5 Field Calculations

You might, for example, write an entry/exit procedure that totals the weights of ordered items and calculates the appropriate shipping charge upon entry in a field. When you want the procedure to check a value that has been entered in a field, the value must be loaded into a variable, not a column. To load the value into a column, use an expression in the form to set the column equal to the variable.

The R:BASE SKIP command is designed for use with entry/exit procedures in forms. Use SKIP to tell R:BASE to skip over or skip back a specified number of fields, or skip to a specified field.

For example, the following entry/exit procedure tells the form to skip to the field with the column amount if there was any value in the variable *vcashtype.*

```
IF vcashtype IS NOT NULL THEN
  SKIP TO amount
ENDIF
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

RETURN

EEP Topics:

EEP Specific Commands
How to Define an EEP
EEP Processing Order
Redisplaying Field Values
RECALC Command Options
EEP Restrictions
EEP Example

1.15.6 RECALC Command Options

The RECALC command, with no options, redisplays the variable in the field that initiated the EEP. By using the RECALC VARIABLES command, all table variables are redisplayed without having to associate them with expressions in the form.

The RECALC TABLES command allows you to immediately display updated values in the lower tables in the form to reflect the actions of the EEP.

You can include both a RECALC VARIABLES and a RECALC TABLES command in your EEP.

EEP Topics:

EEP Specific Commands
How to Define an EEP
Field Calculations
EEP Processing Order
Redisplaying Field Values
EEP Restrictions
EEP Example

1.15.7 Redisplaying Field Values

R:BASE can automatically redisplay a value in a field resulting from an EEP performed on the field. The field must be a variable, and the EEP must be started from the field you want redisplayed.

Include the following steps in the EEP:

- Load the new value into the variable.
- Recalculate the expression with the RECALC command to ensure that expressions based on variables modified by the EEP are always current. RECALC calculates variables dependent on the variable the EEP is associated with.

If you want to load the value into a column, create a form expression that equates the variable and the column, or use the PROPERTY command to return the value to the field.

EEP Topics:

EEP Specific Commands
How to Define an EEP
Field Calculations
EEP Processing Order
RECALC Command Options
EEP Restrictions
EEP Example

1.15.8 Form EEP Processing Order

With the increasing power of EEPs in R:BASE releases, there are several different ways to run an EEP.

This is the order in which the Form EEPs are executed:

1. Custom EEP (embedded)

Copyright © 1982-2024 R:BASE Technologies, Inc.

2. EEP (.eep external file)
3. Stored Procedure
4. Custom Form Action
5. Predefined EEP

Note: The code located within a Custom EEP, or a specified EEP file name would need to be manually removed if one of the lower priority options is subsequently selected as a replacement.

EEP Topics:

EEP Specific Commands
How to Define an EEP
Field Calculations
RECALC Command Options
EEP Restrictions
Redisplaying Field Values
EEP Example

1.16 Environment Optimization

The ability to fine tune R:BASE itself is one of the many enhancements added over the years. Most often, R:BASE is able to make the right decisions about the best way to process commands.

You know your data better and could sometimes pick a more efficient method if given the opportunity. The options detailed below can help you reach maximum performance with R:BASE, but since each application and database is a unique set, only a trial and error process identifies which options help in your situation. Do not forget that different hardware configurations and available memory also affect performance. Experiment with the options and see which ones improve performance for your application. You, the developer, have the opportunity to set R:BASE internal parameters to gain optimum performance for each application.

1.16.1 FASTLOCK

One of the areas of most contention when updating or deleting data is the table lock that must be placed so that R:BASE knows the table is in use and the table structure cannot be modified. Each user modifying data places this lock, and uses the same lock location. With STATICDB set ON, no structure changes are possible on permanent database tables, thus, there is no need for a table lock preventing a user from modifying the structure of a table. Since the lock is not needed, the FASTLOCK setting was added in R:BASE to allow users to turn this table lock off. Commands such as UPDATE, INSERT and DELETE perform much faster with FASTLOCK ON.

The FASTLOCK setting must match for all users of a database and must be set before a database is connected. Users cannot connect a database unless their FASTLOCK setting matches the setting of the open database. FASTLOCK joins STATICDB, MULTI, and TRANSACT as settings that must match for all users connected to a database. FASTLOCK cannot be set ON unless STATICDB is ON. If you SET FASTLOCK ON and connect the database, but STATICDB is OFF, after connecting to the database FASTLOCK is set OFF.

1.16.2 Fast Foreign Keys

To enhance performance when using constraints, primary and foreign keys, the FASTFK setting was added in R:BASE. When using FASTFK, R:BASE does not automatically build an index for foreign keys although indexes are still automatically built for primary keys. With FASTFK, instead of an index, R:BASE simply keeps a counter in file 3, the index file, for the foreign key column. Each foreign key value has a counter keeping track of how many records have this value. This maintains the referential integrity of foreign keys, but does not adversely affect performance by keeping and maintaining an index with an address pointer.

Constraints ensure referential integrity in databases without doing any programming. An index is automatically built and maintained for each primary and foreign defined for a table. Many times, however, foreign keys are used simply to verify that values exist in lookup tables. In a database following good relational design, a table can have many foreign keys, each one of which is indexed. Columns that normally would not be indexed, are now indexed because they are defined as foreign keys.

Copyright © 1982-2024 R:BASE Technologies, Inc.

This affects performance when adding new rows of data, deleting rows or updating data. All the indexes in a table are automatically updated whenever a new row of data is added, or a row is deleted. With updates, when the indexed column changes, its index is updated.

For foreign keys defined to verify lookup values, the lookup table has the corresponding column defined as a primary key which is indexed. That is the index that is used when the form, report, or program does the foreign key lookup. The primary key index is used to verify the data, not the foreign key. Turning the foreign key index off does not affect the lookup speed but dramatically improves the maintenance speed.

For columns that are defined as foreign keys where you need to maintain indexes, simply define a regular, single-column index in addition to the foreign key definition or use the FOREIGN INDEX option of the ALTER TABLE command. CREATE INDEX can be used at any time after a foreign key has been defined when FASTFK is ON. FOREIGN INDEX creates the foreign key and the index at the same time. It cannot be used when FASTFK is ON if the foreign key is already created. FOREIGN INDEX results in a smaller file 3 as only one entry is maintained for the column. When FASTFK is ON, if you define a separate index for a foreign key using CREATE INDEX, file 3 has an entry for the FASTFK counter, and an entry for the index.

To turn FASTFK ON, you must be in single-user mode. Set it to ON, then PACK KEYS to rebuild indexes. The FASTFK setting does not take effect until indexes are rebuilt. The same holds true if you want to turn FASTFK OFF and revert to maintaining true indexes on all foreign keys. After changing the setting of FASTFK, indexes must be rebuilt for the setting to take effect. Make sure the FASTFK setting is set correctly when doing any database maintenance such as PACK or RELOAD.

Two lines display when showing the FASTFK setting, the current setting, and the current setting for the database. The CVAL function returns the current setting for the open database. Below, the settings indicate that the next time indexes are globally rebuilt, FASTFK ON will be used, and foreign keys will not have indexes built for them.

```
R>SHOW FASTFK
(FASTFK   ) OFF Use fast Foreign Key (FK) structures on rebuild
            OFF FASTFK setting for current database
```

R:BASE automatically senses a database's FASTFK setting and connects a user in the correct FASTFK mode. FASTFK ON is stored in the version flag of file 1 of a database. A FASTFK database has a normal version flag of -811 when disconnected. To change the FASTFK setting, you must issue the SET FASTFK command after the database is connected. When a database is initially connected, both settings are always the same.

1.16.3 ISTAT

You can use the ISTAT Function to check the efficiency of settings in order to adjust locking schemes. You should be able to see differences in the results returned by the ISTAT keywords TOTALREADS, TOTALWRITES, and TOTALLOCKS depending on the locking scheme you have set.

1.16.4 PAGELOCK

The PAGELOCK setting is used to turn page locking off and use row locking only. If you know that your application mainly updates or deletes data a row at a time, rather than many rows, SET PAGELOCK OFF for row locking. The valid settings for PAGELOCK are:

- ON - page locking or row locking as appropriate.
- OFF - row locking only, no page locking. R:BASE locks a row, reads the row, makes the change and then releases the row.

The fastest method for updating data in multi-user mode is STATICDB ON, FASTLOCK ON, PAGELOCK OFF. This group of settings results in the fewest contentions between users. PAGELOCK can be changed during an application and can be different for different users of the same database. Otherwise, set PAGELOCK ON for page locking when you are doing an UPDATE and/or DELETE affecting many rows in a table.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.16.5 RULES

If you know that the data being loaded or changed is correct, SET RULES OFF. You can see a small performance increase even if you do not have any rules. If many rules have to be checked, you see a larger performance increase. RULES can be set differently by individual workstations.

1.16.6 MESSAGES

SET MESSAGES OFF eliminates the time required for screen display of messages. This is particularly noticeable if you are doing repetitive UPDATE or INSERT commands. MESSAGES can be set differently by individual workstations.

1.16.7 STATICDB

STATICDB limits structural changes to the database, and activates a read-only schema mode. When STATICDB is ON, permanent changes cannot be made to the database structure (table and column names, for example). New tables can be created, but they are temporary and visible only to the user who created them. By restricting changes to the database structure, R:BASE does not continually re-read the structure to check for changes. Multi-user applications run faster. All users connected to a multi-user database must have the same setting for STATICDB the default is OFF.

1.16.8 MANOPT, #TABLEORDER

Use the MANOPT setting in a multi-table select or with a view to explicitly specify the table order for R:BASE to use when joining tables. Before experimenting with changing the table order, you need to know the default table order, e.g. the table order R:BASE has chosen using its internal optimizing algorithm. To see the table order R:BASE used when joining the tables with SELECT, use the **#TABLEORDER** variable. You must SET DEBUG ON to use MANOPT and #TABLEORDER.

Here is an example using Prodview from the CONCOMP sample database. *Prodview* is a fivetable view. The #TABLEORDER variable shows the table join order and the applicable indexed columns.

```
SET MANOPT OFF -- must be off to use R:BASE's default optimizing algorithm
SET DEBUG ON   -- required to access #TABLEORDER

SELECT * FROM prodview WHERE LIMIT=l
              --viewing 1 row is sufficient to see the table order, and is
fast)

WRITE .#TABLEORDER
customer, transmaster.custid, employee, product, transdetail.transid
```

Results:

table 1 (`customer`) - no index used
table 2 (`transmaster.custid`) - used index column custid to link with customer
table 3 (`employee`) - no index used
table 4 (`product`) - no index used
table 5 (`transdetail.transid`) - used index column transid to link with transmaster

R:BASE looks at the size of the tables (number of rows and columns) and indexes when using the optimizing algorithm to join tables. Removing or adding indexes on linking columns changes the default table order. With *Prodview*, the algorithm determined it was best to start with the Customer table and link to the Transmaster table. R:BASE then takes the result of that join and adds in the Employee and Product tables. Finally, the data in the Transdetail table is looked up using the transid index. Now that the default table order R:BASE used is known, the MANOPT setting is used to change this order and determine performance. The following piece of code checks for performance differences based on the MANOPT setting.

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
SET DEBUG ON
SET MANOPT ON  --or OFF
SET VAR vStart=.#TIME
OUTPUT DUMMY.DAT
SELECT * FROM prodview
OUTPUT SCREEN
SET VAR vEnd = .#TIME
SET VAR vDiff = (.vEnd-.vStart)
SET VAR vTime = (RTIME(0,0, .vDiff))
SHOW MANOPT
WRITE .#TABLEORDER
WRITE .vTime
```

When working with MANOPT, the important part of the view definition or SELECT command is the FROM clause. The table order in the FROM clause of the view definition is transmaster, transdetail, customer, employee, product. With MANOPT ON, R:BASE uses the table order as specified in the FROM clause to join the tables. Notice that the table order in the FROM clause is different from the default table order selected by the query optimizer.

Here is what #TABLEORDER shows with MANOPT ON and selecting from Prodview:

```
transmaster,transdetail.transid,customer.custid, + employee, product. model
```

This is different from the order selected by R:BASE's query optimizer. R:BASE now uses tables in the order specified in the SELECT's FROM clause, incorporating applicable indexed columns. By changing the order of the tables in the FROM clause, you can change the order R:BASE joins the tables for possible improved performance.

In most instances, R:BASE chooses the most efficient way to join your tables; however, you can use the MANOPT setting to experiment and see if a different table order is better.

1.16.9 MICRORIM_FULLOPT

MICRORIM_FULLOPT is a system variable that controls the number of tables that are optomized in a query.

Normally, five tables are optimized in a query. For queries using more than five tables, the MICRORIM_FULLOPT variable can be used to specify the number of tables to optimize.

In general, the R:BASE optimizer makes the correct choice for selecting the fastest method to execute a query, which is that five tables are optimized. Option 2 of MICRORIM_EXPLAIN will force optimization of all tables in a query. MICRORIM_FULLOPT specifies the number of tables to optimize. For queries using over five tables, use MICRORIM_FULLOPT to specify the number of tables to optimize. The more tables that are optimized, the longer the optimization process takes. While you can decrease the data retrieval time by specifying MICRORIM_FULLOPT, you can also increase the optimization time which might counteract the decrease in data retrieval time. As with all optimization techniques, you need to test them with your data.

For queries (SELECTs) using many tables, MICRORIM_FULLOPT specifies the number of tables to be fully optimized. The number MICRORIM_FULLOPT is set to is the number of tables that are fully optimized. Tables above the number specified are only partially optimized. The tables that are optimized are determined by their order in the FROM clause of the query. This variable overrides the optimization part of MICRORIM_EXPLAIN. If MICRORIM_FULLOPT exists, then the 2-bit option of MICRORIM_EPLAIN is ignored.

The command below fully optimizes the first eight tables in a query. If there are more than eight tables in a query, the remaining tables are partially optimized.

```
SET VAR MICRORIM_FULLOPT = 8
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.16.10 MICRORIM_EXPLAIN

The system variable MICRORIM_EXPLAIN shows the decision the optimizer made when executing a particular command. Use this variable to determine if you need to use other optimization methods such as MANOPT. You can see the differences when different indexes or table orders are specified. The optimizer results are placed in the file EXPLAIN.DAT. As with the other variables, MICRORIM_EXPLAIN is initiated by using the SET VAR command. DEBUG must be ON to use MICRORIM_EXPLAIN. For example:

```
SET DEBUG ON
SET VAR MICRORIM_EXPLAIN = 33
SELECT * FROM prodview
RBEDIT EXPLAIN.DAT

SelectCost=1  (OptimizationTime=0ms)
  ProdView Sequential

SelectCost=1.429222  (OptimizationTime=0ms)
  Product Sequential
  InvoiceDetail (ColumnName=Model,Type=F) Random Dup=18 Adj=0.9513889
  InvoiceHeader (ColumnName=TransID,Type=P) Random Dup=1 Adj=1
  Customer (ColumnName=CustID,Type=P) Random Dup=1 Adj=1
  Employee (ColumnName=EmpID,Type=P) Random Dup=1 Adj=1
```

MICRORIM_EXPLAIN shows the optimizer selections for both implicit and explicit selects. In the above command, the data from the view, prodview, is retrieved sequentially, but the join used to create the data for prodview uses indexes. The optimizer table order and index use matches that returned by #TABLEORDER. The values for "Dup" and "Adj" are from the SYS_DUP_FACTOR and SYS_ADJ_FACTOR columns in the SYS_INDEXES table.

The available options for MICRORIM_EXPLAIN are:

1	Output the table order.
2	Force optimization of joins over five tables. Seven table joins take about 5 seconds to optimize, an eight table join takes approximately 8*5 second to optimize etc. With joins over five tables, the optimizer does not try all possible combinations but picks an order based on the "best" indexes. This option is ignored if MICRORIM_FULLOPT is set.
4	Output the command file name and next byte offset (location of command within the file). Commands from the R> Prompt are so noted.
8	Output the current date and time.
32	Display the sort technique used.

To use multiple options, add the options numbers together and set the variable MICRORIM_EXPLAIN to the result. For example:

```
SET VAR MICRORIM_EXPLAIN = (1 + 4 + 8)
SET VAR MICRORIM_EXPLAIN = 13
SET VAR MICRORIM_EXPLAIN = (13 + 32)
```

Output specified by each of the options is put into the file EXPLAIN.DAT. EXPLAIN.DAT is an ASCII file. You can write comments to the file EXPLAIN.DAT. To put information about the command and the time it took to execute in the file, enter these commands after the command is executed:

```
OUTPUT EXPLAIN.DAT APPEND
WRITE 'this is the command ' , .vTime
WRITE ' '
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

OUTPUT SCREEN

This places the execution time and explanation of the command in the file with the optimization selection. It lets you easily test options and see the differences. Delete the file EXPLAIN.DAT to start over. When MICRORIM_EXPLAIN is set, every query made by R:BASE sends output to the file EXPLAIN.DAT. For example, when you use a form to edit data, or a report to print data, output is sent to the file EXPLAIN.DAT because the data for the form or report is retrieved via an internal SELECT command.

The output displays the optimization time in milli-seconds (OptimizationTime=930ms). This time is how long it took R:BASE to determine the best table and index order. Each table order and index the optimizer checks is given a value (SelectCost=9.796483). The lowest value or select cost determines which query order is used. There are four types of access methods in MICRORIM_EXPLAIN that can be used to retrieve the data:

Sequential	No index was used. The rows are retrieved from the table in the order they were entered.
Random	An index access method.
Very Random	An index access method.
IndexOnly	No data was retrieved from file 2, all the data was retrieved from file 3.

When index access is used (IndexOnly, Random, Very Random), the column name or index name and the index type are displayed. For example:

```
transmaster (ColumnName=empid, Type=F) IndexOnly Dup=3.8 Adj=1
transmaster (IndexName=dateindex,Type=I) Random Dup=1.055556 Adj=1
```

The index types are:

P	Primary Key
F	Foreign Key
I	Index

In addition, the terms "PreSort" and "SortMerge" are used. In multi-table joins, a small table may be placed entirely in memory and sorted there-the table is pre-sorted. Lookups are then done from the pre-sorted table in memory to the other tables in the join. PreSort is used when the sort column(s) is in only one table. The sort column(s) is not a linking column. When this method is used, PreSort shows in the EXPLAIN.DAT output. SortMerge is used only if PreSort is also used. The SortMerge method is most efficient if the tables are not indexed. When PreSort and SortMerge are indicated, you also see SortRemoved. The sort has already been done, and is not repeated again for the final output. Sorting the final output is not included in the SelectCost. For example:

```
SelectCost=O.3035209 (OptimizationTime=Oms) SortRemoved
  Prodlnfo Sequential PreSort (DATA_ONLY)
  Customer (ColumnName=CustID,Type=I) Random Dup=1 Adj=1
  Products (ColumnName=ProdID,Type=I) Random Dup=1 Adj=1
```

The 32-bit option of MICRORIM_EXPLAIN can be used to display the sort strategy R:BASE used for the query. For example:

```
SortStrategy = DB_TAG (internal=1)
SortStrategy = DATA_ONLY (internal=2)
```

The possible sort strategy options are:

DB_TAG (Internal=1)	The sort is done using the rowid value. Only the columns specified in the ORDER BY are used to sort the data. Lookups are then done from the ordered list of rowid values to retrieve the rest of the data for display. When there are many rows of data, or many columns to display, this is the type of sort used.
DATA_ONLY (Internal=2)	The sort is done using all of the data to be displayed. An entire row of data is sorted together. This type of sort is used when

Copyright © 1982-2024 R:BASE Technologies, Inc.

	there is a small number of rows or only a few columns to display.
FILE_TAG (Internal=3)	The sort is done to a temporary file. Multi-table queries involving NOTE fields and sorts on views usually use this type of sort.
DB_TAG_PRESORT (Internal=4)	The sort is done using the rowid of the column(s) specified in the ORDER BY. The sort columns must be from just one table in a multi-table SELECT. The table is presorted, then lookups are done to retrieve the rest of the data.
FILE_TAG_PRESORT (Internal=5)	The sort is done to a temporary file. The table is presorted, then lookups are done to retrieve the rest of the data.

1.17 File Extensions

In general, the following R:BASE file extensions have been used:

File Extension	Description
ALL	R:BASE for Windows table structure and data unload file, possibly accompanied by a corresponding .LOB file
APP	ASCII application file, typically generated by Application Express in all R:BASE versions up to 6.5++
API	R:BASE for DOS application File, typically generated by Application Express, for internal use, in all R:BASE versions up to 6.5++
APW	R:BASE for Windows application file, typically generated by Application Express, for internal use, in all R:BASE versions up to 6.5++
APX	Binary application file created using CodeLock, also generated by Application Express in all R:BASE versions
ASC	ASCII text file
B01, B02, etc.	Before image file, which contains a log of the commands issued in a transaction. The before image files are in the same directory as the database and have extensions that begin with "B," such as DATABASE.B01.
CFG	ASCII configuration file
CMD	Command file (not used anymore due to the now .BAT file extension used in the latest Windows operating systems)
CSV	Comma separated value data file
DAT	Application startup file, or text data file
DBG	Save Watch Variable list from the Trace Debugger
EEP	Entry/Exit Procedure file (used in Forms)
FRM	R:BASE for Windows form unload file, accompanied by a corresponding .LOB file
INI	ASCII initialization file containing configuration settings
LOB	Binary or large object data created during an unload process for objects stored in the fourth database file
LBL	R:BASE for Windows label unload file, accompanied by a corresponding .LOB file
MNU	ASCII menu file, referenced in APP files
PLY	Play back file, traditionally used with R:BASE versions up to 6.5++
PRC	ASCII stored procedure file
PRN	ASCII text printout file
PRO	Unloaded stored procedure, or a compiled procedure file, traditionally compiled using CodeLock
RB0	Diagnostic file when database structure issues exist
RB1	The file extension for a database created using R:BASE versions 4.5 up to R:BASE X.5. This file contains the database structure.
RB2	The file extension for a database created using R:BASE versions 4.5 up to R:BASE X.5. This file contains the data for the database.
RB3	The file extension for a database created using R:BASE versions 4.5 up to R:BASE X.5. This file contains the database indexes.
RB4	The file extension for a database created using R:BASE versions 5.0 up to R:BASE X.5. This file contains binary and text large objects.

Copyright © 1982-2024 R:BASE Technologies, Inc.

RBA	Windows application file, typically generated by the Application Designer in R:BASE versions 7.x and higher
RBC	R:Charts chart file
RBF	The file extension for databases created using R:BASE 3.0, R:BASE for DOS, and R:BASE System V.
RBL	R:BASE Plugin file for R:BASE 7.x and Turbo V-8
RBM	R:BASE Module file (R:BASE Plugin file for version 9.0 and higher)
RBS	File extension for databases created with R:BASE 5000 and R:BASE 4000.
RCP	R:Compiler project file
RFF	R:BASE external form file, or an R:Fax file
RGW	R:BASE Gateway import/export specification file
RMD	R:BASE ASCII command file
RMT	R:Mail Editor template file
RPT	Windows report unload file, accompanied by corresponding .LOB file
RSF	R:Synchronizer script file
RTB	R:Backup script file
RX0	Diagnostic file when database structure issues exist
RX1	The file extension for a database created using R:BASE Turbo V-8, R:BASE eXtreme 9.5 (64), R:BASE X.5 Enterprise, and R:BASE 11. This file contains the database structure.
RX2	The file extension for a database created using R:BASE Turbo V-8, R:BASE eXtreme 9.5 (64), R:BASE X.5 Enterprise, and R:BASE 11. This file contains the database data.
RX3	The file extension for a database created using R:BASE Turbo V-8, R:BASE eXtreme 9.5 (64), R:BASE X.5 Enterprise, and R:BASE 11. This file contains the database indexes.
RX4	The file extension for a database created using R:BASE Turbo V-8, R:BASE eXtreme 9.5 (64), R:BASE X.5 Enterprise, and R:BASE 11. This file contains binary and text large objects.
SCH	R:BASE syntax schema file
TBL	R:BASE for Windows table structure/data unload file
VIE	R:BASE for Windows view structure unload file

Please Note: No user is required to follow or adhere to all of the above file extensions, as some exceptions can be made.

1.18 Foreign Data Sources and ODBC

You can use ODBC to link to other data sources. Ordinarily, you can work with only one R:BASE database at a time. With ODBC, R:BASE can access tables from other databases. This means that you can use ODBC to access multiple R:BASE databases at once. For example, you can take information from one R:BASE database and use it in another. You can also open an R:BASE database, then access databases from other programs, such as Oracle or Paradox.

Also, ODBC allows the developer to write one application that has the ability to access data from databases created with different ODBC-compliant products. For example, you could write an application that would select data from SQL Server, Oracle, Paradox, and R:BASE databases, separately or all at the same time, depending on which database drivers you have installed.

Basically, a data source would need be created for the database product that you will to have access to. If you want R:BASE to have access to SQL Server data, you would set up a SQL Server data source. If you are providing access to R:BASE data, you would set up a R:BASE data source.

ODBC Topics:

Copyright © 1982-2024 R:BASE Technologies, Inc.

See also:

SATTACH Command
SCONNECT Command
SDETACH Command
SDISCONNECT Command
SET PASSTHROUGH Command
SSQL Command

1.18.1 How ODBC Works

When you access other databases, you are working with the actual data, not a copy. Not only can you view information from other databases, but you can modify and add information, depending on the access rights you have. For example, from R:BASE, you can add a row of data to a Paradox database.

In order for different database programs to communicate with each other, they use a standard Structured Query Language, or SQL. This language allows R:BASE to work with data in foreign databases. R:BASE does not actually perform actions itself, but sends the equivalent SQL command to the foreign database. The foreign database receives the command and performs the action on its own data. The foreign database controls the data validation, not R:BASE. For example, you are working in R:BASE and connect to a foreign database. If you add a row of data to a foreign table, R:BASE sends an SQL INSERT command to the foreign database, which then validates the data and adds the row to its table. You should be familiar with the access rights and constraints associated with the attached tables in their host systems.

ODBC Topics:

ODBC Compliance
Setting Up Data Sources
Connecting Data Sources and Tables
Working with Data Sources
Disconnecting Data Sources and Tables

See also:

SATTACH Command
SCONNECT Command
SDETACH Command
SDISCONNECT Command
SET PASSTHROUGH Command
SSQL Command

1.18.2 ODBC Compliance

If you are planning on using ODBC with R:BASE, you must make sure that your character settings are set correctly by performing the following:

1. Go to the R> Prompt
2. Connect to your database
3. Type: `SHOW CHAR`
4. Compare your values to the chart below

Character	Set To	Description
MANY	%	Percent Sign
SINGLE	_	Underscore
QUOTES	'	Single Quote

5. If they do not match, change them by typing SET TYPE=CHAR
   (where TYPE is from the "Character" column and CHAR from the "Set To" column)

Copyright © 1982-2024 R:BASE Technologies, Inc.

For example, SET MANY=%

ODBC Topics:

How ODBC Works
Setting Up Data Sources
Connecting Data Sources and Tables
Working with Data Sources
Disconnecting Data Sources and Tables

See also:

SATTACH Command
SCONNECT Command
SDETACH Command
SDISCONNECT Command
PASSTHROUGH Setting
SSQL Command

1.18.3 Setting Up Data Sources

To access a data source, you must first install the corresponding driver. When you install R:BASE, the installer loads the ODBC driver necessary for connecting to other R:BASE databases.

To set up a data source, you can use the ODBC Data Source program, which is installed with R:BASE, or you can use the ODBC Data Source Administrator program located in the Windows Control Panel, under the "Data Sources (ODBC)" icon. This may be located within "Administrative Tools".

To install drivers other than the R:BASE driver, refer to their respective installation manuals.

ODBC Topics:

How ODBC Works
ODBC Compliance
Creating an R:BASE Data Source
Connecting Data Sources and Tables
Working with Data Sources
Disconnecting Data Sources and Tables

See also:

SATTACH Command
SCONNECT Command
SDETACH Command
SDISCONNECT Command
SET PASSTHROUGH Command
SSQL Command

1.18.3.1 Creating an R:BASE Data Source in R:BASE

To create a data source in R:BASE, you can use the ODBC Data Source program. With R:BASE launched while connected to your R:BASE database, proceed through the following instructions:

1. From the main Menu Bar, select "Utilities" > "Connect SQL Data Source". This will open the "Data Source" dialog. You can also enter SCONNECT at the R> Prompt to achieve the same result.
2. Click on the "Machine Data Source" tab.
3. Select the "New" button.
4. Select "System Data Source"
5. Choose the R:BASE 11 Database Driver, then the "Next" button.
6. Then click on the "Finish" button. You'll now see
7. Create a data source name specific to the database.
8. Enter the database path or use the "Browse" button to locate the database path (specify the RX1 file only).

Copyright © 1982-2024 R:BASE Technologies, Inc.

9. The "Description" is optional.
10. When you click OK, you will be taken back to the ODBC Data Source window. The listing of Data Sources should include the DSN you have just created.

1.18.3.2 Creating an R:BASE Data Source via Control Panel

To create an R:BASE data source using the Windows Control Panel, proceed through the following instructions:

When creating a new DSN (Data Source Name), you can do so within the ODBC Data Source Administrator program.

1. To access the ODBC Data Source Administrator, open the Windows Control Panel, and select the "Data Sources (ODBC)" icon. This may be located within "Administrative Tools".
2. You need to determine if the application you are working with requires a User or System DSN. Once you decide, select the appropriate tab, then click the "Add" button.
3. Next, select the database driver. Select the R:BASE Database Driver, and click the Finish button.
4. The R:BASE DSN setup screen is displayed where you must the appropriate information. In some cases, you will need to restrict the name of the Data Source if your development environment does not allow certain characters. The "Full Path of Data Source" should include the RX1 file of the R:BASE database.
5. Click OK to save the DSN information.

At the ODBC Administrator window, the list of Data Sources should include the newly created DSN.

1.18.4 Connecting Data Sources and Tables

When R:BASE attaches a foreign table, R:BASE only includes those columns in the table that have recognizable data types. Also, if the foreign tables have names that are illegal in R:BASE, or if a foreign table name is the same as one in the R:BASE database, R:BASE prompts you to specify an alias that will be used for the foreign table while that table is attached.

To ensure that data in attached tables is updated correctly, attached tables should have a primary or unique key defined. If no primary or unique keys are defined, R:BASE prompts for the columns you want to use to determine unique rows.

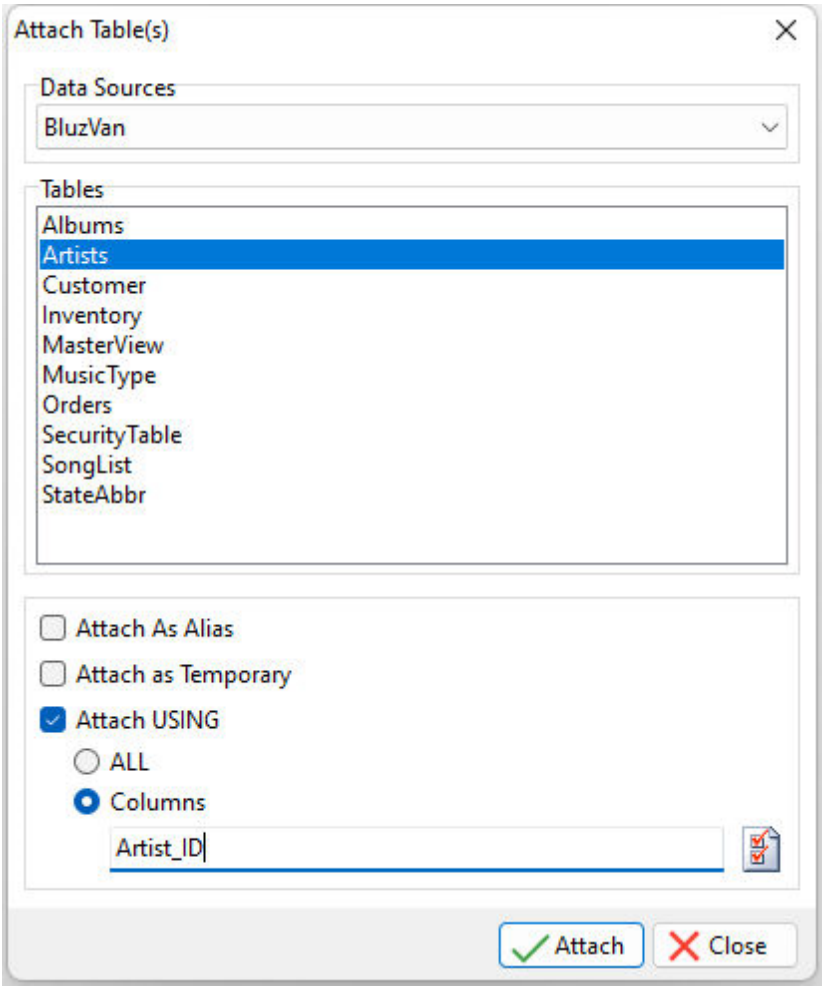
Attaching a Foreign Database Table

1. Connect to your database which foreign tables you want moved into.
2. From the main Menu Bar, select "Utilities" > "Connect SQL Data Source". This will open the "Select Data Source" dialog. You can also enter SCONNECT at the R> Prompt to achieve the same result.
3. Click on the "Machine Data Source" tab.
4. Select the Data Source Name from the list, and press the "OK" button.

   At this point, you can select "Utilities" from main Menu Bar and notice that the options for "Disconnect SQL Data Source" and "Attach SQL Database Tables" are enabled. This means you have correctly connected to the data source.

5. From the main Menu Bar, select "Utilities" > "Attach SQL Database Tables". You can also enter SATTACH at the R> Prompt to achieve the same result. The following dialog will be displayed.

Copyright © 1982-2024 R:BASE Technologies, Inc.



6. From the displayed dialog, choose your defined Data Source from the drop down box to preview the list of available database tables to attach.
7. Select the table you want to attach.

   Notice the option for "Attach As Alias." It may be necessary for you to assign an alias name if the foreign data source table does not follow the same table name restrictions as R:BASE. The option for "Attach as Temporary" is also available to attach a foreign data source as a temporary table. The "Attach USING" option is available to assign the QualKey columns. Either all columns are assigned, or the button may be used to specify the column(s) that uniquely identify the rows.

8. Press the "Attach" button.
9. Now, if you navigate to the Database Explorer, you will see the table(s) listed.

When foreign data source tables are attached to R:BASE, the tables will appear differently than other tables in the Database Explorer. The foreign tables will display a green icon next to the name whereas regular R:BASE tables will display a blue icon. Also, the word "SERVER" will appear under the "Rows" column and the table comment will include "Server table with the attached table name.

You can repeat the steps for any other database/tables you wish to attach to the main database.

If needed, use the PROJECT command to add the table permanently to the database.

ODBC Topics:

How ODBC Works
ODBC Compliance

Copyright © 1982-2024 R:BASE Technologies, Inc.

Setting Up Data Sources
Working with Data Sources
Disconnecting Data Sources and Tables

See also:

SATTACH Command
SCONNECT Command
SDETACH Command
SDISCONNECT Command
SET PASSTHROUGH Command
SSQL Command

1.18.5 Working with Data Sources

To work with a data source, you must first open a database in R:BASE, then you can connect the data source and select the table(s) with which you want to work. You can attach as many tables as you like.

Once you have attached a table from a data source, you can work with the data as you would with R:BASE data. However, you cannot use the following R:BASE commands with attached data source tables:

ALTER TABLE	AUTONUM
BACKUP	CREATE INDEX
DROP COLUMN	DROP INDEX
INTERSECT	JOIN
RENAME COLUMN	RENAME TABLE
SUBTRACT	UNION
UNLOAD DATA FOR tblname AS ASCII	UNLOAD DATA FOR tblname AS CSV

Restrictions on attached tables by foreign databases remain in effect--you can only perform actions on attached tables that are allowed within the restrictions of any access rights, rules, or constraints on the foreign database. For example, if your access rights to a foreign database allow you to view data only, you will receive an error message from the data source if you try to modify data.

For best performance, your attached tables should have a primary or unique key defined.

ODBC Topics:

How ODBC Works
ODBC Compliance
Setting Up Data Sources
Connecting Data Sources and Tables
Disconnecting Data Sources and Tables

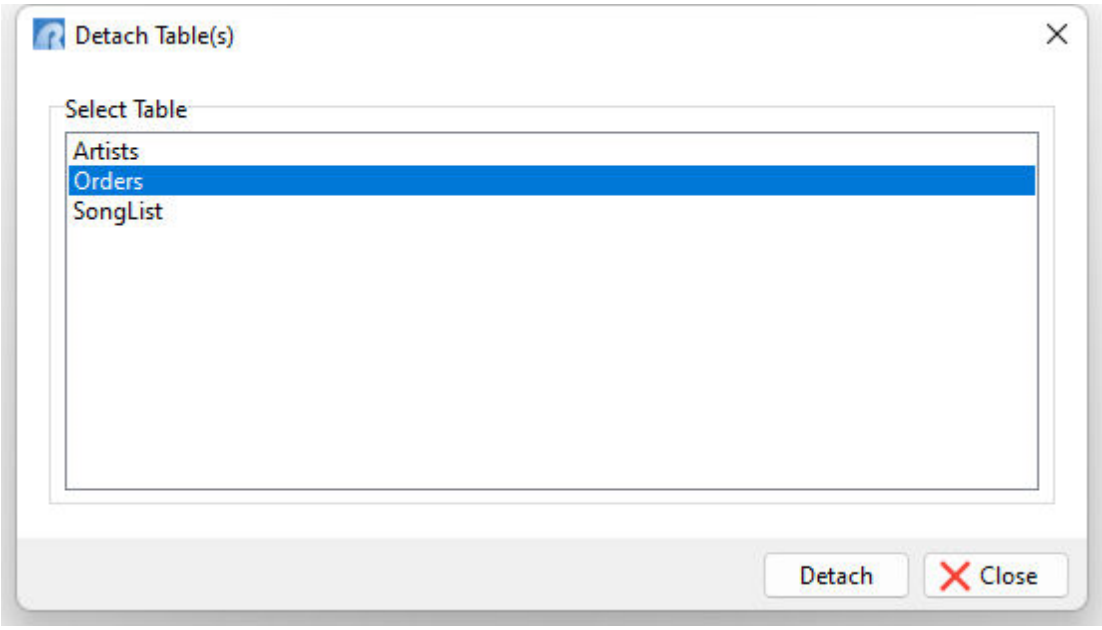
See also:

SATTACH Command
SCONNECT Command
SDETACH Command
SDISCONNECT Command
SET PASSTHROUGH Command
SSQL Command

1.18.6 Disconnecting Data Sources and Tables

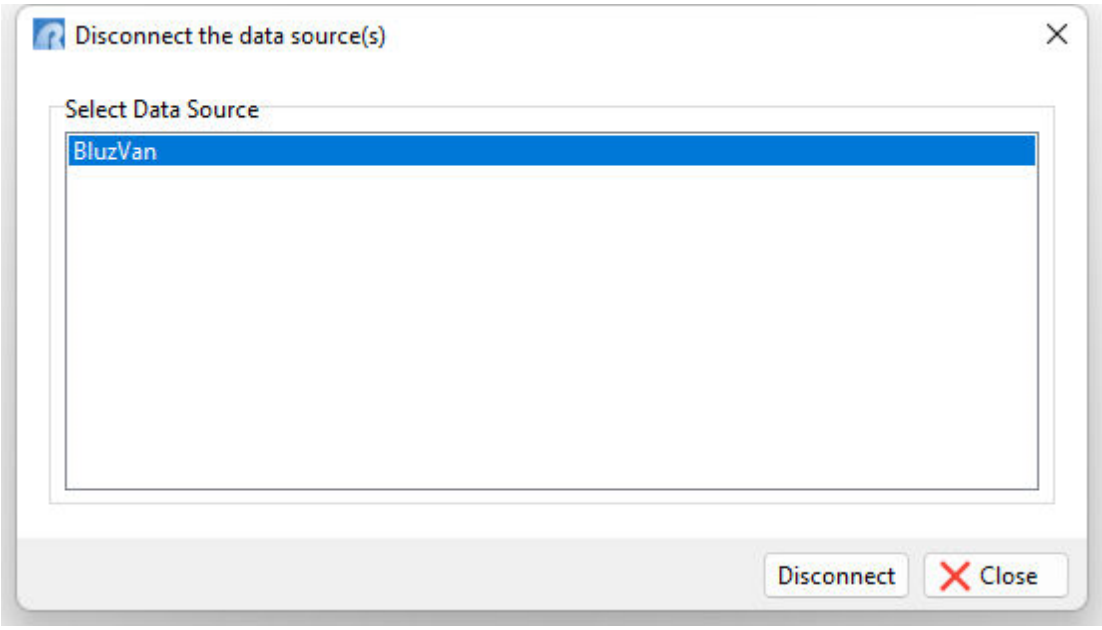
Detaching an SQL Database Tables
To detach an SQL Database Table, select "Utilities" > "Detach SQL Database Tables" from the main Menu Bar. By doing so, the following dialog will be displayed.

Copyright © 1982-2024 R:BASE Technologies, Inc.



You can also enter SDETACH at the R> Prompt to achieve the same result.

Disconnecting an SQL Data Source
To disconnect an SQL Data Source, select "Utilities" > "Disconnect SQL Data Source" from the main Menu Bar. By doing so, the following dialog will be displayed.



You can also enter SDISCONNECT at the R> Prompt to achieve the same result.

ODBC Topics:

How ODBC Works
ODBC Compliance
Setting Up Data Sources

Copyright © 1982-2024 R:BASE Technologies, Inc.

Connecting Data Sources and Tables
Working with Data Sources

See also:

SATTACH Command
SCONNECT Command
SDETACH Command
SDISCONNECT Command
SET PASSTHROUGH Command
SSQL Command

1.18.7 About the SYS_ROWVER Column

If you have ever used Oterro to generate or alter tables you may have noticed that those tables get a new column added to the end; the SYS_ROWVER column.

The SYS_ROWVER column is a computed integer column that contains the version of the row. Its formula is as follows:
(IFNULL((SYS_ROWVER+1),0,(SYS_ROWVER+1))).

This translates to the following: If the existing SYS_ROWVER is null then the row version is one. Otherwise increment the row version by one.

This is used, in most cases, by outside programs that do not have access to the internal R:BASE concurrency system (Table Locking, Row Locking, Page Locking, Column vs Row Verification and so on.) For this reason, Oterro defaults to AUTOROWVER ON and R:BASE defaults to OFF. When AUTOROWVER is ON any CREATE TABLE or ALTER TABLE adds the SYS_ROWVER column to the table being altered or created.

Examples

For example, let's assume you were working with an application that connected to R:BASE via Oterro. You might see...

```
SEL ROWID,FNAME,SYS_ROWVER +
   INTO vROWID,vFNAME,vROWVER+
   FROM TABLENAME WHERE CRITERIA
```

Something happens where the user edits the data...

```
SEL SYS_ROWVER INTO vNEWVER +
   WHERE ROWID=vROWID

IF vNEWVER = vROWVER THEN
   UPDATE TABLE +
      SET COLUMN FNAME = vFNAME +
      WHERE ROWID = vROWID
ELSE

   DIALOG 'The row you are editing has been changed by another user.
Continue?' +
      vRESP vEND YES

   --Continue with code that either processes the update anyways or that
   --allows end user to resolve the conflict or that does something else.

ENDIF
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.18.8 ODBC System Variables

The following are ODBC system variables. If a server error occurs and the following variables exist, they will be filled in with the information about the error. If an ODBC variable contains multiple errors, the last error will be first. Each ODBC variable is limited to 4096 characters.

ODBC_DELIMITER
The system variable that controls whether the ODBC system variables contain a single error or multiple errors.

If the ODBC_DELIMITER variable exists, is not null, and contains at least one character, then the other ODBC system variables will contain all previous errors separated by the first character of this variable, otherwise they will only contain the last error.

ODBC_ERRORMSG
The system variable that stores the ODBC error messages as Text.

If a server error occurs and the ODBC_ERRORMSG variable exists, it will be filled in with error message text from the server. If the ODBC_DELIMITER variable has a value this will contain multiple errors.

ODBC_SQLSTATE
This TEXT variable is a five character code defined by ODBC.

If a server error occurs and the ODBC_SQLSTATE variable exist, it will be filled in with a five character error code defined by ODBC. If the ODBC_DELIMITER variable has a value then the ODBC_SQLSTATE variable will contain multiple error numbers.

ODBC_NATIVEERROR
This TEXT system variable is the ODBC error number from the server.

If a server error occurs and the ODBC_NATIVEERROR variable exists, it will be filled in with the error number from the server. If the ODBC_DELIMITER variable has a value, then the ODBC_NATIVEERROR variable will contain multiple error numbers.

Define the variables using the SET VARIABLE command as follows, prior to use with ODBC tables.

```
SET VAR ODBC_DELIMITER TEXT = '_'
SET VAR ODBC_ERRORMSG TEXT
SET VAR ODBC_SQLSTATE TEXT
SET VAR ODBC_NATIVEERROR TEXT
```

1.19 Hot Keys

Within the various modules of R:BASE there are hot keys that allow easier use with the program. The following hot keys are supported throughout the entire R:BASE environment:

Common Functionality:

[Ctrl+A]	Select All
[Ctrl+C]	Copy
[Ctrl+V]	Paste
[Ctrl+X]	Cut
[Ctrl+Z]	Undo

Display and Navigation:

[F1]	Main R:BASE Help
[F3]	Data Dictionary

Copyright © 1982-2024 R:BASE Technologies, Inc.

[Ctrl+F3]	Help Index
[Ctrl+F4]	Closes an R:BASE module window
[Ctrl+E]	R:BASE Editor
[Ctrl+L]	Database Explorer
[Ctrl+R]	R> Prompt
[Ctrl+W]	Watch Variables
[Ctrl+F11]	Scratch Pad
[Ctrl+Alt+M]	Magnifying Glass
[Ctrl+Tab] or [Ctrl+F6]	Next window for R:BASE module
[Ctrl+Shift+Tab]	Previous window for R:BASE module
[Shift+F1]	Help File for current R:BASE module
[Shift+F4]	Tile Windows
[Shift+F5]	Cascade Windows
[Shift+F9]	Help Search

Database Explorer Search

[Ctrl+F]	Searches for text in Form, Report, Label, Application, and External Form **Custom EEPs**
[Ctrl+L]	Searches for caption/label text in Form, Report, Label, and External Form **Control Properties**
[Ctrl+Shift+F]	Searches for text in Form, Report, and Label **Expressions**
[Ctrl+N]	Searches for names, comments, driving tables, etc. for Database Explorer objects

Forms Runtime:

[F2]	Add Row
[F4]	Lowers the List for Drop-Down Controls
[F5]	Reset Field
[F7]	Previous Row
[F8]	Next Row
[F9]	Delete Row
[Ctrl+D]	Multi-Column Sorting (Enhanced DB Grid)
[Ctrl+F]	Record Search (Enhanced DB Grid, Tree View)
[Ctrl+L]	Display Filer (Enhanced DB Grid)
[Ctrl+S]	Show/Hide Columns (Enhanced DB Grid)
[Ctrl+F10]	Refresh Row
[Ctrl+Alt+I]	Provides form information including name, driving table, slave tables, database (with path), and current folder
[Ctrl+Alt+Shift+I]	Performs an image capture of the current form at runtime and launches the BLOB Editor to possibly alter (image annotations) and save
[Ctrl+Shift+D]	Clear Sorting (Enhanced DB Grid)
[Shift+F3]	Pop-up Menu
[Shift+F7]	Previous Table
[Shift+F8]	Next Table

Report/Label Preview:

[Ctrl+Alt+I]	Provides report information including name, driving table, slave tables, database (with path), and current folder

Copyright © 1982-2024 R:BASE Technologies, Inc.

[Ctrl+Alt+Shift+I]	Performs an image capture of the current report/label at runtime and launches the BLOB Editor to possibly alter (image annotations) and save
[Shift]+Mouse wheel scroll	Moves page to page in the Print Preview screen

The following modules offer many additional Hot Keys which are specific to their individual environment:

- R:BASE Editor
- Interactive Debugger
- Data Browser
- Database Explorer
- R> Prompt
- Form Designer
- Report Designer

1.20 Icons

The following icons can be used with the DIALOG and PAUSE commands.

Icon "*value*" Parameter	Icon
APPS	
ATTENTION	
CONFIRM	
ERROR	
HELP	
INFO	
QUESTION	
SERIOUS	
STOP	
WARNING	
WINDOWS	

1.21 Image Annotations With the BLOB Editor

You can add, edit, or delete images (BLOBs) within your database files. The R:BASE BLOB Editor has also been enhanced to manage multipage images. Drawing objects, like lines, boxes, text, etc., can also be added to images.

When adding or displaying images in the R:BASE BLOB Editor, annotations can be made using several type of drawing tools. Objects include:

Button	Description

Copyright © 1982-2024 R:BASE Technologies, Inc.

	Draw Line
	Draw Box
	Draw Ellipse
	Draw Text
	Draw Ruler
	Draw Polyline
	Draw Angle

Image Annotation Notes:

- After an object is added to the image, the properties will automatically display. Double clicking on the object will redisplay the Properties dialog.
- The object(s) added become part of the image.
- When a field object is selected, small tan circles appear on the corners and sides or line ends, based upon the drawing object added. These circles are called handlebars which are used to manipulate the size and location.
- Objects can be stretched, resized and/or manipulated by hovering the mouse cursor over and dragging any of the available handlebars that are displayed upon the object's edges or end points. The cursor will change to a double pointed arrow.
- Objects can be moved by hovering the mouse cursor along any line portion of the object between the handlebars and dragging the mouse cursor. The cursor will change to a black four-pointed cross.
- After placing an object, any objects added after will use the same pen and brush properties, if the setting applies.
- When placing a text object, "Draw Text" option, on an image and then clicking the "OK" button to save the object properties, start typing your text. You will notice the text being displayed against a grey background.
- In order to save an image annotation, double click on the thumbnail (left side of the separator). The change will then be visible in the thumbnail image.
- Selecting the "OK" or "Cancel" buttons (lower right corner) of the R:BASE BLOB Editor will save or cancel your changes accordingly.

Copyright © 1982-2024 R:BASE Technologies, Inc.

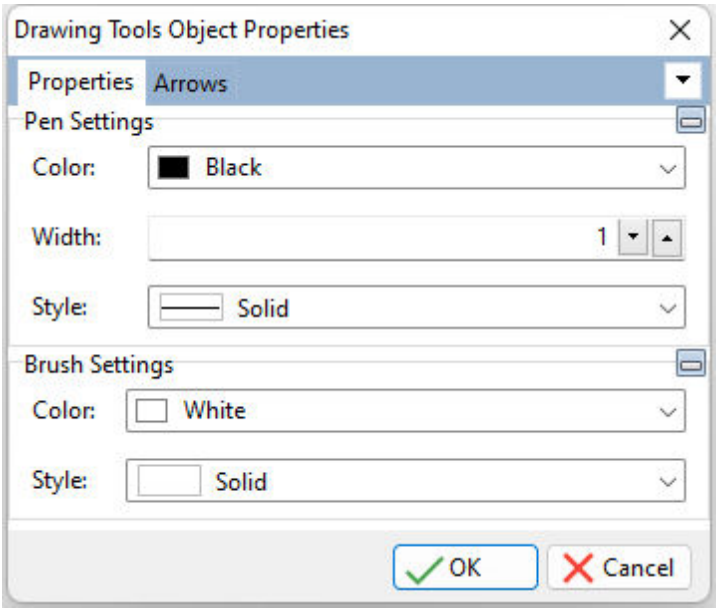
1.21.1 Line

Pen Settings

Color - specifies the line color
Width - Specifies the line width
Style - specifies the line style

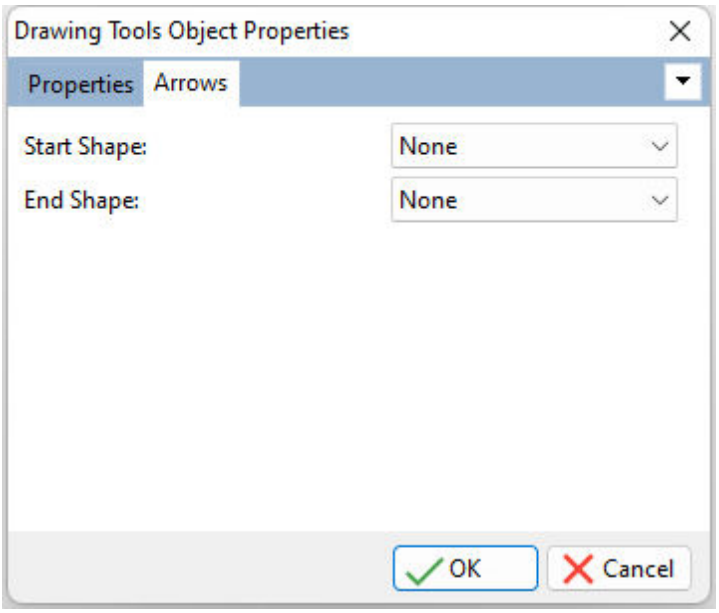
Brush Settings

Color - specifies the fill color for the arrow start and end shapes, if defined
Style - specifies the fill style for the arrow start and end shapes, if defined



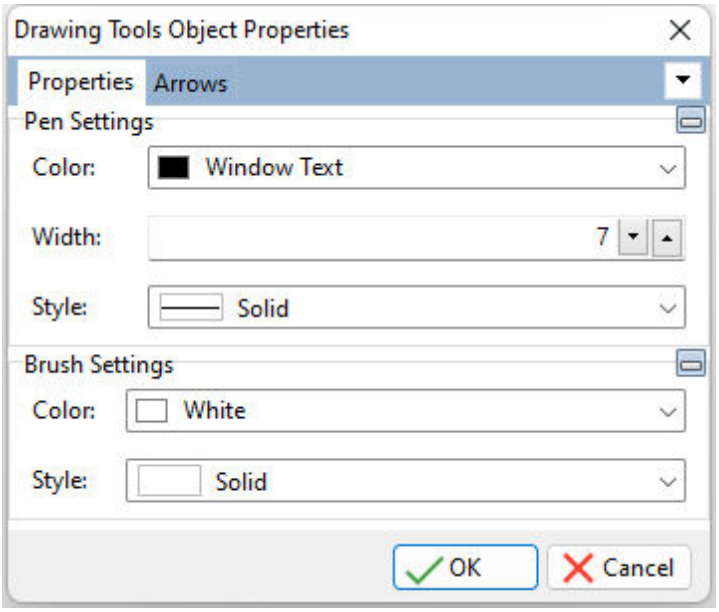
Start Shape - specifies the arrow start shape for the line

End Shape - specifies the arrow end shape for the line



The following object properties for a line were used to add the line object in the screen shot below.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Notice the handlebars on each end of the line that can be used to stretch or move the starting/ending points for the arrow.

Copyright © 1982-2024 R:BASE Technologies, Inc.

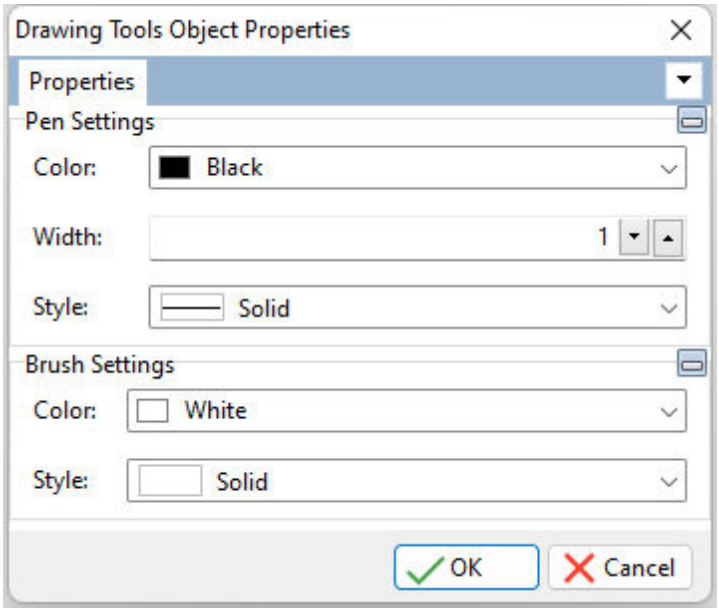
1.21.2 Box

Pen Settings

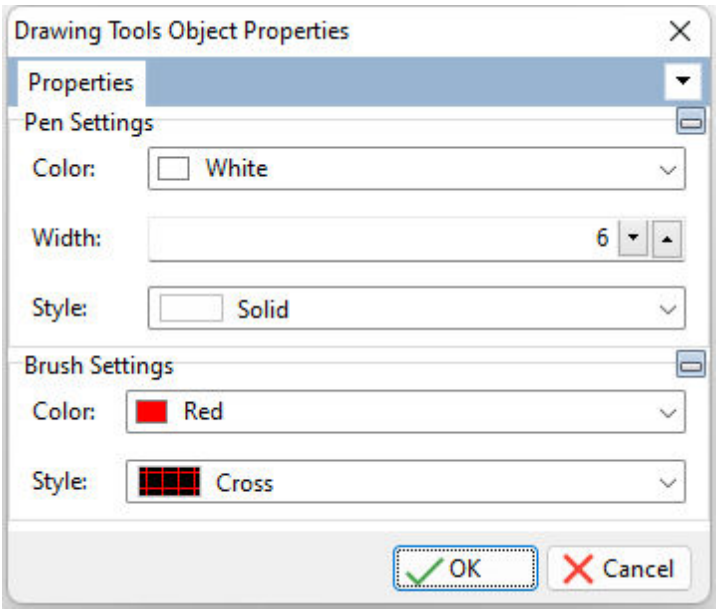
Color - specifies the box border color
Width - Specifies the box border width
Style - specifies the box border style

Brush Settings

Color - specifies the box fill color
Style - specifies the box fill style



The following object properties for a box were used to add the object in the screen shot below.



Notice the handlebars on each corner, and between each corner, that can be used to stretch or move the box.

Copyright © 1982-2024 R:BASE Technologies, Inc.

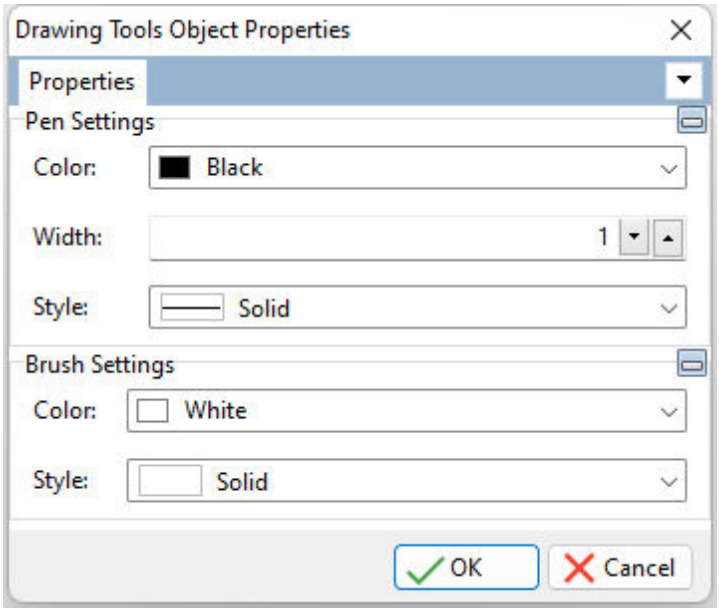
1.21.3 Ellipse

Pen Settings

Color - specifies the ellipse color
Width - Specifies the ellipse width
Style - specifies the ellipse style

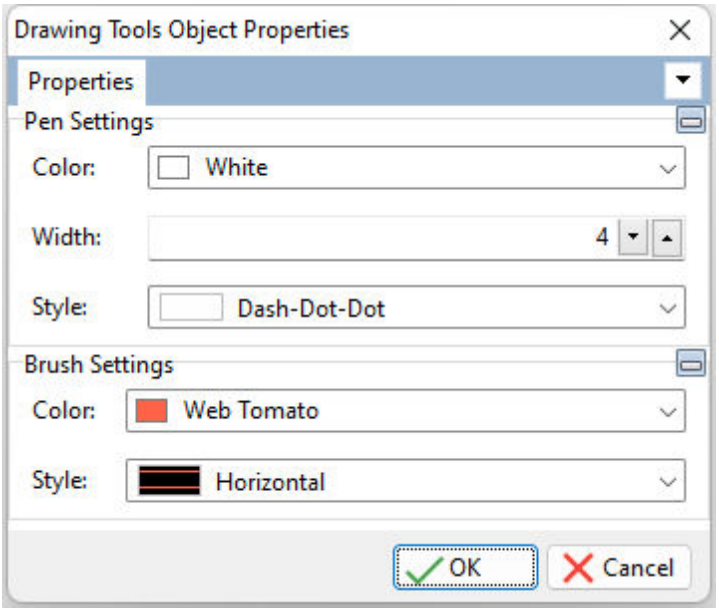
Brush Settings

Color - specifies the ellipse fill color
Style - specifies the ellipse fill style



The following object properties for an ellipse were used to add the object in the screen shot below.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Notice the handlebars at the corners and sides that can be used to stretch or move the ellipse.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.21.4 Text

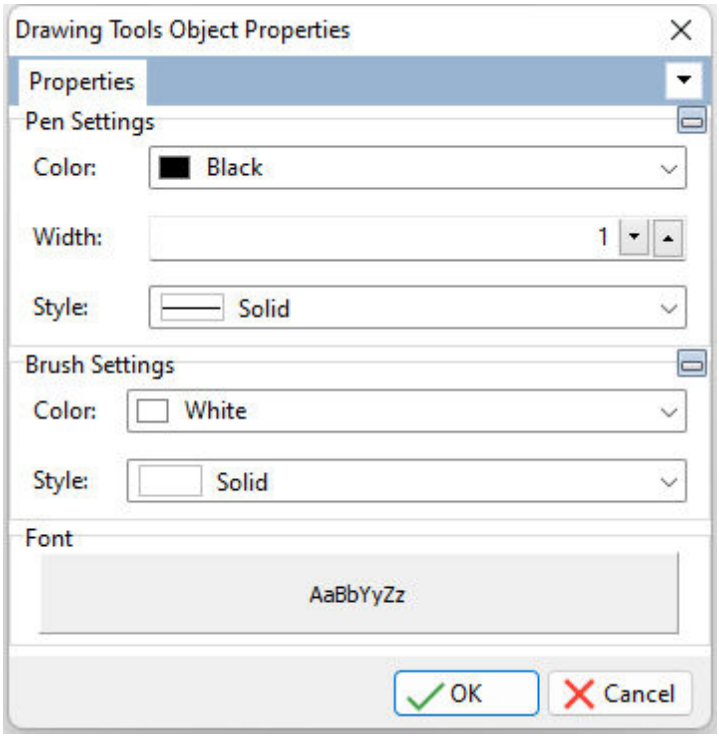
Pen Settings

Color - specifies the text and border color
Width - Specifies the text border width
Style - specifies the text border style

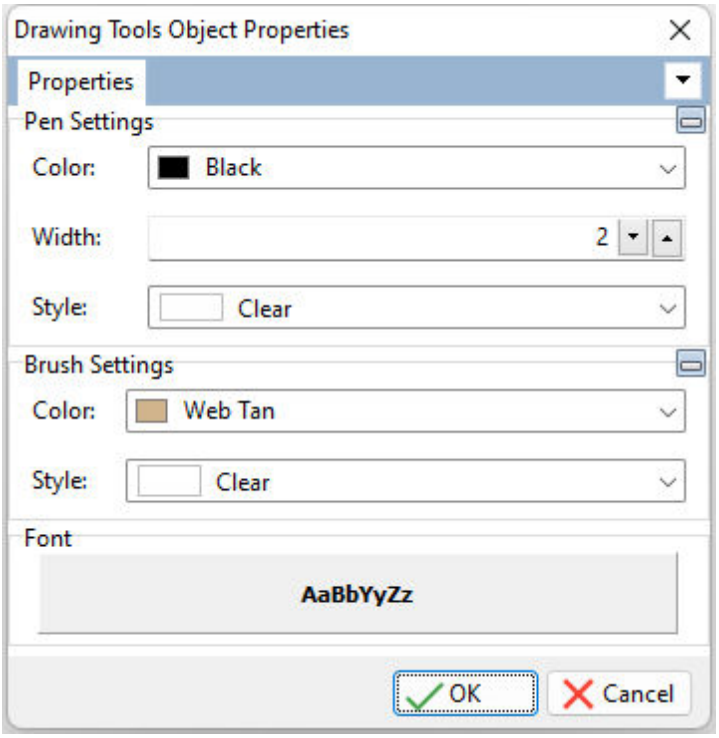
Brush Settings

Color - specifies the text background color
Style - specifies the text background style

Font - specifies the text font type, size, and style



After adding the Text object, the following object properties were used.



Copyright © 1982-2024 R:BASE Technologies, Inc.

The gray box is the input mode waiting for your text key entry.

With the above settings, the text is displayed in the screen shot below. Notice the handlebars at the corners and sides that can be used to stretch or move the text object.

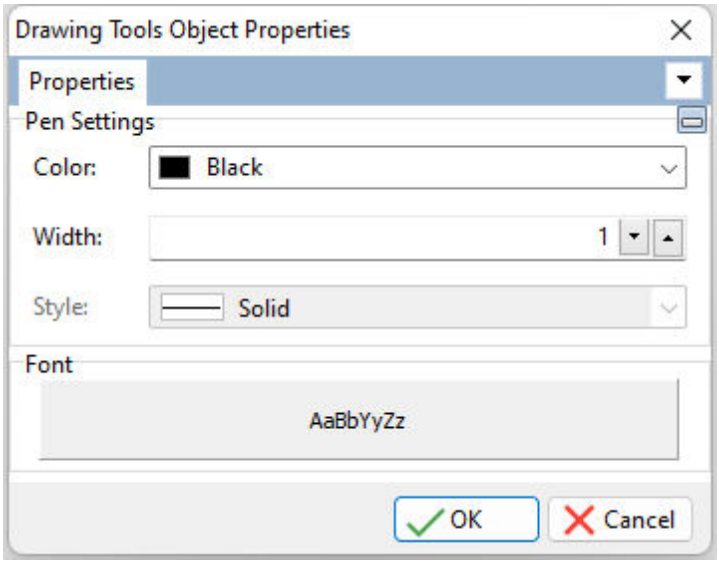
Copyright © 1982-2024 R:BASE Technologies, Inc.

1.21.5 Ruler

Pen Settings

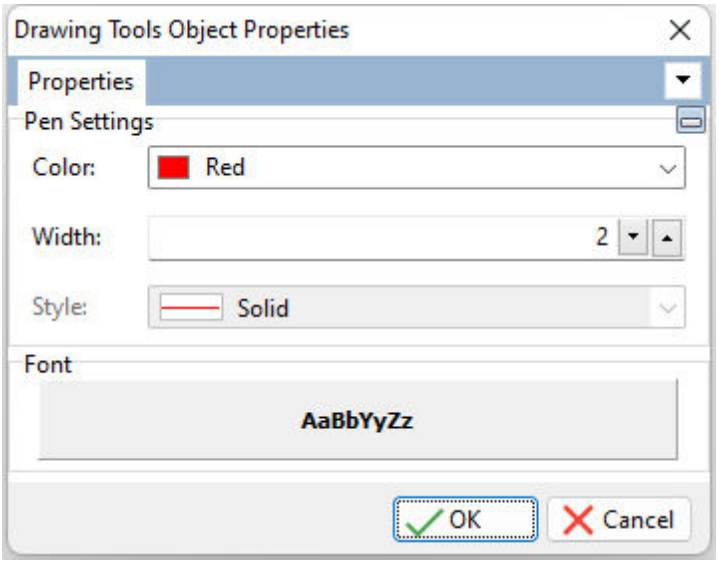
Color - specifies the ruler line color
Width - Specifies the ruler line width

Font - specifies the text font type, size, and style

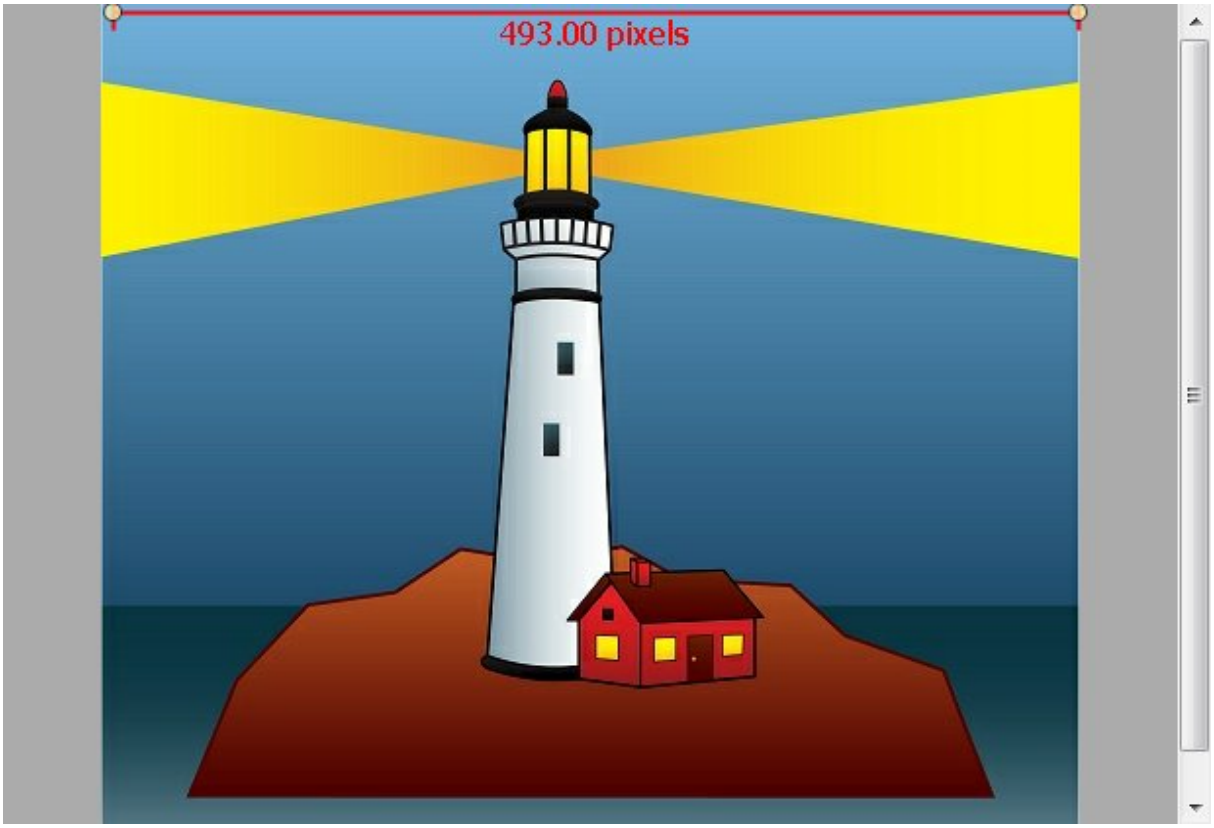


After adding the Ruler object, the following object properties were used.

Copyright © 1982-2024 R:BASE Technologies, Inc.



With the above settings, the ruler is displayed in the screen shot below.

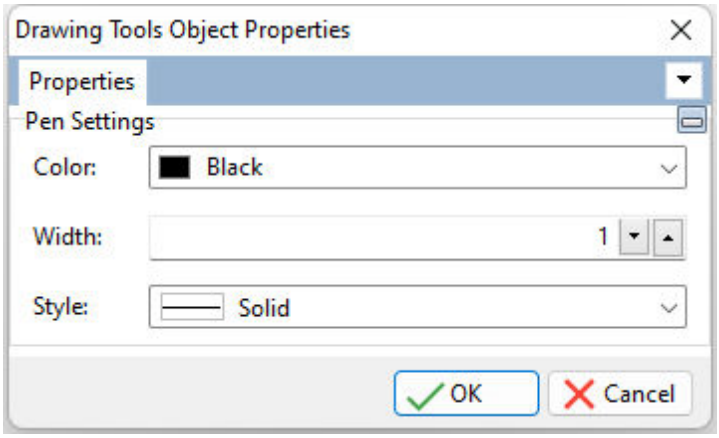


Copyright © 1982-2024 R:BASE Technologies, Inc.

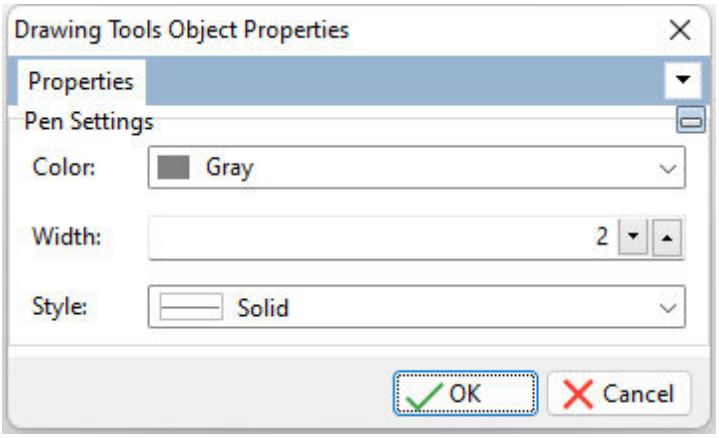
1.21.6 Polyline

Pen Settings

Color - specifies the polyline border color
Width - Specifies the polyline border width
Style - specifies the polyline border style



The following object properties for a polyline were used to add the object in the screen shot below.



Notice the handlebars at the corners and sides that can be used to stretch or move the polyline.

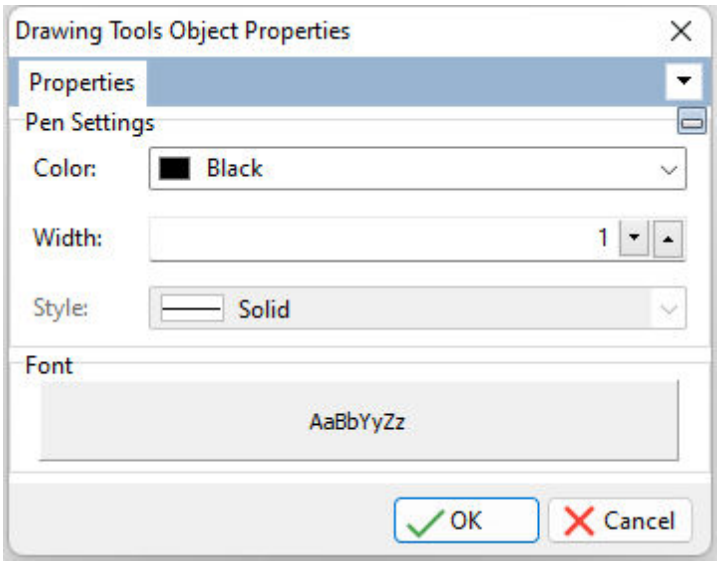
Copyright © 1982-2024 R:BASE Technologies, Inc.

1.21.7 Angle

Pen Settings

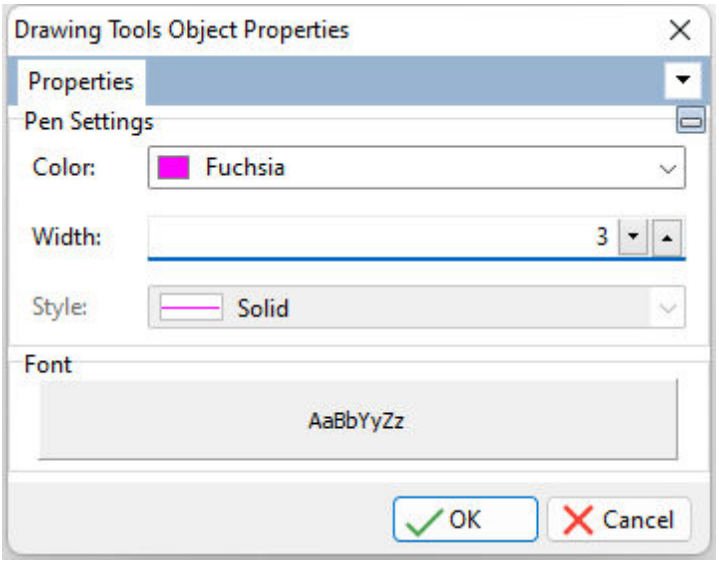
Color - specifies the angle line color
Width - Specifies the angle line width

Font - specifies the text font type, size, and style

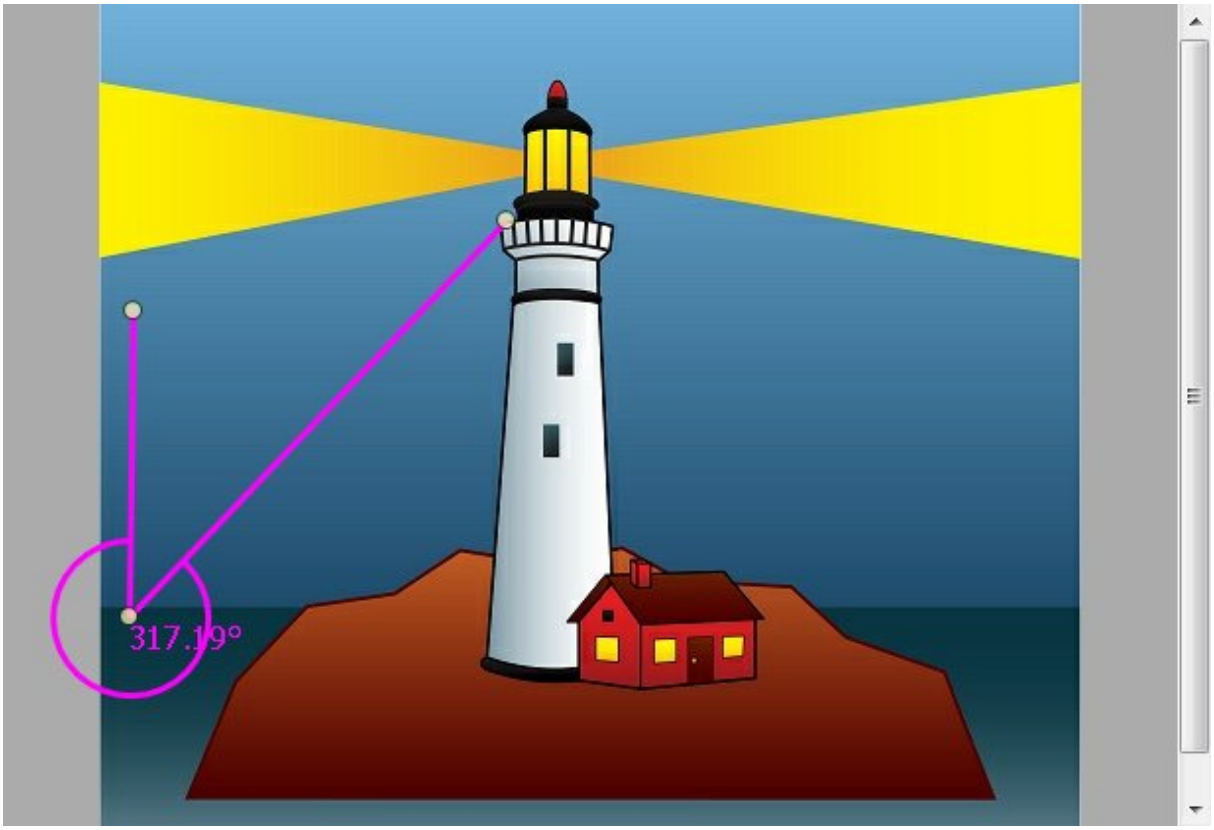


After adding the Angle object, the following object properties were used.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Notice the handlebars at the end and corner that can be used to stretch or move the angle object.



Copyright © 1982-2024 R:BASE Technologies, Inc.

1.22 Indexes

An index provides a pointer to the location of a column value in each row of a table, which allows R:BASE to search for rows of data much faster than searching rows sequentially. In general, R:BASE processes an operation that contains an indexed column faster than it processes one without.

The R:BASE index is similar to an index in a book; both indexes allow you to find information faster. Instead of searching through a book page by page, you can look up the topic in the index and find the exact page number of the topic. Similarly, you can apply indexes to columns so that R:BASE finds data faster.

When you apply an index to a column, R:BASE records the location of every value in that column. Then, when you look for or sort information in the column, R:BASE uses the index to find the rows you need quickly. For example, you want to list the bonuses that employee 102 earned; if the *EmpID* column in the *SalesBonus* table is indexed, R:BASE finds and searches that column faster. Indexes are most useful when you have tables with many rows.

This means that by indexing the appropriate columns, you can speed up your applications. Processes that formerly took 20 minutes or more may be completed in only a few seconds. An index on an R:BASE column speeds up access to a row of data in much the same way that an index in a book speeds up access to a page.

To understand how indexes increase processing speed, it's helpful to know how R:BASE stores a database. An R:BASE database consists of four files; each file has a different file extension - RX1, RX2, RX3, and RX4 for R:BASE 11. File 1 contains the definition of the database structure; file 2 contains the data in the database; file 3 contains the indexes to the data stored in file 2, and file 4 contains the large binary object data (LOB). LOBs include graphic files or large text data. R:BASE also stores forms, reports, and labels as LOBs in File 4.

When you include an indexed column in a WHERE clause, R:BASE uses the index in file 3 to find the location of each row in the table in file 2. Without an indexed column, R:BASE must sequentially search each row in that table in file 2 to find the data. By sorting an index list, you give R:BASE nearly instantaneous access to the specific index reference to a column name.

An indexed column improves the performance of the following commands, clauses, or operations:

- CREATE VIEW
- DELETE DUPLICATES
- INTERSECT
- JOIN
- Lookups in forms or reports
- PROJECT
- RULES
- SELECT (when it includes a WHERE clause)
- SUBTRACT
- UNION
- VIEWS
- WHERE...

1.22.1 Choosing the Columns to Index

It's a very good idea to put an index on key columns and on linking columns. A key column is a column that uniquely identifies rows. A linking, or common column, is a column that exists in two or more tables in order to establish a relationship between the tables. In effect, by choosing to link the common columns between two tables by adding a primary key and foreign key relationship, the columns are automatically indexed. You can also add a Unique key to a column, which is also automatically indexed. Primary keys, foreign keys, and unique keys are all types of constraints, which specifically control the data that enters your database by applying powerful data-integrity rules. By applying a constraint to a column, you can prevent irreconcilable and empty data from being entered add at the same time add an index.

Aside from constraints, an index can be added to a column for faster data retrieval in cases where the columns are not used in a primary key/foreign key relationship, and the column is likely to be used as part of a WHERE Clause whose values are at least moderately unique. In some cases, an index can be

Copyright © 1982-2024 R:BASE Technologies, Inc.

applied to columns that have RULES applied to them, which allows R:BASE to check the RULE faster. An index can also be added to the linking columns in views. An indexed column can contain null values, but R:BASE uses an index most efficiently if each row in the indexed column contains a value.

You can apply the following types of indexes:

- **Unique index** - Ensures that the values entered in the indexed column are unique
- **NOT NULL Index** - The index is populated with references to rows where the column value is not NULL. NULL values in the column are not added to the index. Not NULL indexes are ideal for columns which contain a considerable amount of NULL values, and where the not-NULL data is ideal for index retrievals. If a WHERE clause includes "ColName IS NULL" then the index is not used.
- **Full or partial text index** - For columns with NOTE or TEXT data types. R:BASE preserves each character in the indexed column (a full text index). Or, you can specify the number of characters to preserve, and R:BASE hashes (converts characters to a 4-byte integer) the remaining characters (a partial index).

  To keep the index file from becoming too large, use a partial index--specify enough characters to guarantee the values are unique. If the preserved values are not unique, R:BASE must unhash the values before it can identify the rows, which slows performance. If you do not specify the number of characters to preserve, R:BASE preserves all of them, unless there are more than 200 characters defined; then, R:BASE preserves the first 32 and hashes the rest.
- **Multi-column index** - A combination of up to 8 columns in one index. For example, if you consistently search three columns when working with a certain database, you can define a separate index for each column. Or, you can define one index for all three columns. R:BASE searches a multi-column index faster than three separate indexes. Multi-column indexes should be constructed in the same order as used in the WHERE Clause (that is to mix ASC and DESC in the index based on actual use). With a query upon rows by date and listing the results in descending order, so the most current invoices appear first, it is important to define the index on the table's date column in descending order as well. In a situation where you need to use more than one column to uniquely identify a row, try combining them into a computed column and then indexing the computed column.

Primary, unique, and foreign key columns are indexed automatically. In addition, the following types of columns are good for indexing:

- Columns that are neither primary nor foreign keys, but are frequently referred to in queries and sorts.
- Columns that have rules applied to them--in most cases, R:BASE can use indexing to check rules faster.
- Linking columns in views.

Although indexes speed up searches, they may slow down data entry for 2 reasons:

- they occupy space on the disk in the number 3 database file
- it takes time to build the index for each value as it is entered

Therefore, try to limit the use of indexes.

1.22.2 Assigning and Removing an Index

You can add an index to or remove an index from a column at any time by using either the Data Designer (RBDefine) or the CREATE INDEX or DROP INDEX commands. When you use the database-building menus to define a table, you can index columns as you define them. When you define a table using CREATE TABLE, you must add the index after you define the table.

Valid names must start with a letter, and can include the following characters:

- Letters (A-Z)
- Numbers (0-9)
- **#** (pound sign)
- **_** (underscore)
- **$** (dollar sign)

Copyright © 1982-2024 R:BASE Technologies, Inc.

- **%** (percent sign)

To speed up some operations, remove the indexes from the columns before the operation and then rebuild the indexes after the operation finishes. For example, you can load a large number of rows into a table without RULES by using the LOAD and INSERT commands or by using GATEWAY to import data. First, you remove the indexes from the columns in the table, then load the records, and finally rebuild the indexes. This method speeds up processing because all of the data writes to file 2 occur at the same time, followed by all of the index writes to file 3.

When dropping and creating indexes constantly, it is recommended that you maintain the index file and perform a PACK on the indexes. Use the PACK KEYS in single-user mode with MULTI set to OFF, or PACK INDEX in a multi-user mode with MULTI set to ON.

1.22.3 Optimizing Indexes

You can create optimal indexes by paying attention to the data type of the indexed column and by minimizing the number of duplicate values in the column. The more frequently a particular value occurs, the less efficient that column becomes as an indexed column.

The fastest, most efficient data types for indexed access are INTEGER, REAL, DATE, TIME, and TEXT with a defined length of four characters or less.

You may find that indexing a table with 1500 or fewer rows actually increases the total time to search for rows of data, since the task also includes the time it takes to build the index.

1.22.4 Indexing Long TEXT Values

Because the defined length of TEXT values is often more than four characters, R:BASE must hash (convert) the TEXT values containing more than four characters to create a four-byte index called a hash value. Even though a TEXT column may contain unique text values, it may generate duplicate hash values.

How R:BASE Hashes TEXT Data

R:BASE attempts to create unique hash values by performing the following operations:

- Fills the first and second bytes of the four-byte hash value with the binary equivalents of the first and second characters in the TEXT string.
- Sums the binary values of all the odd-numbered characters in the TEXT string beginning with the third. Then it divides the sum by 256 and uses the remainder to fill the third byte of the hash value.
- Sums the binary values of all the even-numbered characters in the TEXT string beginning with the fourth. Then it divides the sum by 256 and uses the remainder to fill the fourth byte of the hash value.

If two hash values created by hashing two different TEXT strings are identical, R:BASE builds a multiple occurrence table in file 3 to store the duplicate hash values.

This is why TEXT strings that share only their beginning two characters (often the case with part numbers and with form, report, and label names) may have the same hash value, thereby reducing the effectiveness of the index. If you can ensure that the first two characters of each TEXT column value are unique, you can improve the performance of an index on that TEXT column, because you'll eliminate all of the multiple occurrence tables.

Another way to make hash values unique is to define indexed TEXT columns with as few characters as possible while remembering to make the column values unique, especially with regard to the first two characters.

IHASH to the Rescue

If the first two characters of TEXT values are identical, which is often true with part numbers, such as AB-100, AB-101, AB-102, and so on, try using the IHASH function and a computed column to reduce the likelihood of duplicates. It's important to understand that IHASH doesn't guarantee a unique hash value.

Copyright © 1982-2024 R:BASE Technologies, Inc.

IHASH may actually hash two different TEXT values to the same hash value (index). What IHASH does is to create more non-duplicate values, though it doesn't guarantee that all values will be unique.

How to Use IHASH

IHASH is a conversion function, not a command. Its syntax is simple:

```
(IHASH(arg,n))
```

Arg can be a TEXT column, a dotted variable, or a value. The width, n, tells R:BASE how many characters, starting with the first, are needed to establish a unique string. If n is zero, R:BASE uses the entire length of the string.

The first consideration in using IHASH is to decide whether to use it at all. For example, IHASH may not improve performance when used on a TEXT column that holds unique text names of four characters or less. But it will improve performance on a TEXT column that has many values with identical first and second characters. It all depends on whether IHASH can reduce the number of identical values in the index.

IHASH Step by Step

Here's an example that uses IHASH to speed up indexed access and joins by increasing the speed of a key, indexed TEXT column named *partid.* Follow these steps:

1. Define an INTEGER computed column, including the IHASH function:

   ```
   ALTER TABLE tblname ADD hashid = (IHASH(partid,0)) INTEGER
   ```

   If *partid* exists in other tables, you may want to add *hashid* to them too. For example, if *partid* links a parts table with an *invoice* table, use these commands to add *hashid*:

   ```
   ALTER TABLE parts ADD hashid = (IHASH(partid,0)) INTEGER
   ALTER TABLE invoice ADD hashid
   ```

2. Drop the index from *partid* and index the computed column *hashid*:

   ```
   DROP INDEX partid IN tblname
   CREATE INDEX ON tblname hashid
   ```

   Index *hashid* in all the tables where it appears.

3. Use the newly indexed computed column in WHERE clauses and multi-table SELECT commands, but be sure to continue to include the actual TEXT column also. For example, use it to find a specific part number instead of using this WHERE clause:

   ```
   command ... WHERE partid = .vpartid
   ```

   Use this SET VAR and WHERE clause:

   ```
   SET VAR vhash = (IHASH(.vpartid,0))
   command ... WHERE (hashid = .vhash AND partid = .vpartid)
   ```

   You must use the dotted variable (.*vhash*) instead of directly using the expression in the WHERE clause, because R:BASE doesn't use an index if the WHERE clause uses an expression.

   You must include AND *partid* = . *vpartid* to ensure that you get the right value. Remember, there's no guarantee that the IHASH value will be unique.

   If you use the computed IHASH column (hashid) in a multi-table join, you must also remember to include both conditions in the WHERE clause. For example, the following rule checks for a unique value:

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
RULES 'Value must be unique.' +
FOR tblname SUCCEEDS +
WHERE partid IS NOT NULL +
AND NOT EXISTS +
(SELECT partid FROM tblname t2 +
WHERE t2.hashid = tblname.hashid +
AND t2.partid = tblname.partid)
```

Correlated sub-SELECTs force R:BASE to do an internal join using indexes.

Here's another example of a multi-table join using *hashid*:

```
SELECT t1.partid, SUM(t2.price) +
FROM parts t1, invoices t2 +
WHERE t2.hashid = t1.hashid +
AND t2.partid = t1.partid +
GROUP BY t1.partid
```

Speeding Up IHASH

When you use IHASH, try not to use zero as the width (n). Zero causes R:BASE to use the entire string. For example, if you're sure that the first 10 characters in a TEXT 50 column are enough to establish it as unique, use 10 rather than zero with IHASH.

1.22.5 Using WHERE Clauses with Indexes

A WHERE clause can be in a command, a rule, or a lookup expression. If a WHERE clause has only one condition, and if the conditional operator following the indexed column is =, BETWEEN, or IS NULL, R:BASE uses the index on the column.

The BETWEEN operator doesn't use the index on a data type that must be hashed. So if the operator is BETWEEN, the data type of the indexed column must be DATE, TIME, INTEGER, REAL, or TEXT with a defined length of 4 or less.

R:BASE doesn't use the index if the WHERE clause contains a wildcard character or an expression. To make R:BASE use the index, replace expressions with constants or dotted variables and get rid of the wildcards.

When a WHERE clause contains more than one condition and all conditions are combined with AND, the clause must have at least one indexed column that uses =, BETWEEN, or IS NULL if you want R:BASE to use indexes. Under these conditions, R:BASE chooses the condition that places the greatest restriction on the WHERE clause for the indexed search. R:BASE uses the first of two conditions when both are at the same level of restriction. Here are the three levels of restriction:

- = is most restrictive
- IS NULL is less restrictive
- BETWEEN is least restrictive

(R:BASE does not use indexes on BETWEEN when the command allows data modifications.)

1.22.6 Using ORDER BY with Indexes

You can significantly reduce the time R:BASE takes to process an ORDER BY clause when the column or columns listed in the ORDER BY clause are included in an index with the same column sort order as that specified in the ORDER BY clause.

An example would be if you are processing data by the invoice date and listing the results descending order, so the most current invoices appear first, you would have an index on the table's invoice date column that was created to process the data in the descending order as well.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.22.7 Using Index-Only Retrieval

If it can, R:BASE will do an index-only retrieval. This is the fastest method of retrieving data. When the columns selected for display are limited to the column or columns in the index used in the WHERE clause, R:BASE will retrieve the data as it reads the index information from file 3. It does not need to look at the data stored in file 2. Index-only retrieval is done only when the columns to be retrieved are all included in the index. If the table is small, however, index-only retrieval may not be faster. As a rule of thumb, R:BASE will choose index-only retrieval if the length of the index columns is less than 50% ofthe row length (in bytes). If the table is small (not many columns), other retrieval methods are usually faster than index-only.

The SELECT COUNT(*) command uses index-only retrieval if there is an indexed column in the table. Instead, use a SELECT COUNT(colname) command, where the column specified in the command is not an indexed column, and the column has a value for every row. If the column is indexed, R:BASE will use index-only retrieval. If the column contains nulls, those rows are not counted. Do not use this command to check for broken pointers or to compare performance with other commands.

1.22.8 Indexing Computed Columns

Adding indexes can speed up your applications in several ways. One method that allows very quick data retrieval involves creating a computed column of unique values based on one or more columns.

For example, here is part of a program that helps in the scheduling of flight simulators for pilot training. A simulator code (*scode*) in combination with a date (*sdate*) uniquely identifies each row in the table. To make WHERE clauses fast, the following indexed computed INTEGER column (*sindex*) contains this expression:

```
sindex = (JDATE(sdate) + (scode * 100000)) INTEGER
```

The application prompts for a simulator code and date, which it puts in two variables: *vscode* (INTEGER) and *vsdate* (DATE). Then to quickly find the row, the application uses the following code (replace "command" with any command that uses a WHERE clause):

```
SET VAR vsindex = (JDATE(.vsdate) + (.vscode * 100000))
command ... WHERE sindex = .vsindex
```

The above code proved twice as fast as this slower alternative:

```
command ... WHERE scode = .vscode AND sdate = .vsdate
```

The more rows you have, the greater the efficiency.

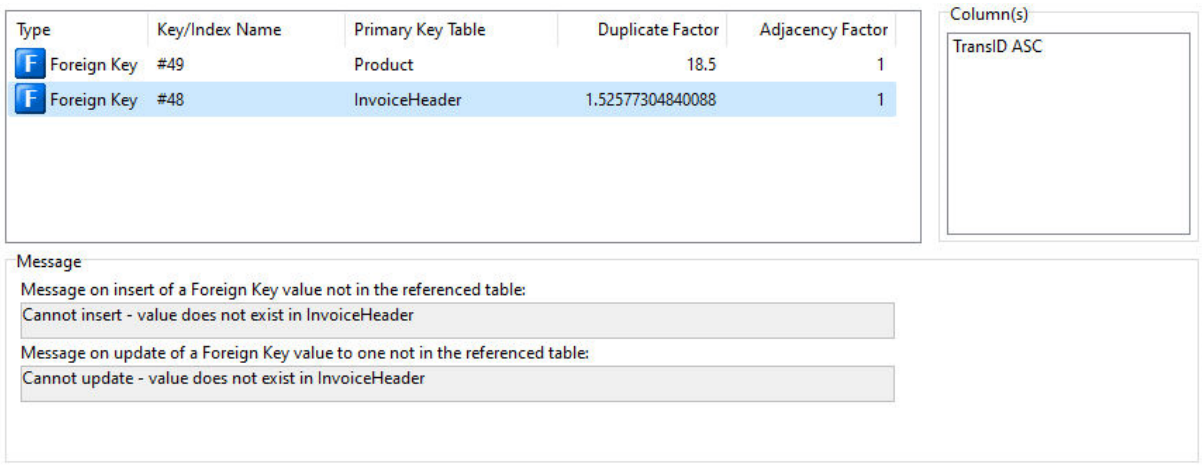
1.22.9 Index Efficiency

To assist in determining the efficiency of indexes, check the **Duplicate Factor** and **Adjacency Factor** values within the Data Designer.

For Duplicate Factor, this is the average number of times each value appears in the column. The number 1 means that values are never duplicated, or unique. The "higher" the number, the less efficient the index.

For Adjacency Factor, this number is the estimate of the probability that two rows with similar index values will be physically located together in the #2 data file. A "higher" number means more efficient retrieval when reading rows in index order.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Duplicate Factor is the computed duplicate factor, used by R:BASE during retrieval to guess the fastest way to find the result set. This number is the average number of times each value appears in the column. A number of 1.0 means that values are never duplicated, or always unique. Zero means that the value is unknown. The higher the number, the less efficient the index.

Adjacency Factor is the computed adjacency factor, used by R:BASE during retrieval to guess the fastest way to find the result set. It is the estimate of the probability that two rows with similar index values will be physically located together in the .RX2 file. A higher number means more efficient retrieval when reading rows in index order. Zero means that the value is unknown.

The values within these two columns are computed when indexes are rebuilt or reloaded, for example, during a PACK or RELOAD.

It is recommended that you reconsider the use of indexes with a high Duplicate Factor value. But, there is not an exact Duplicate Factor benchmark for which to warrant its value "too high". It's value is calculated based upon the number of duplicate values that are within the column, which would be directly related to how many rows are in the table.

A Duplicate Factor with a value of 500 within a table containing 750 rows would be an example of a poorly implemented index. On the other hand, a Duplicate Factor of 500 within a table containing 785,000 rows would be considered productive. The Duplicate Factor value is used by R:BASE to guess the fastest way to find your results. The recommendation to reconsider the use of indexes with a high Duplicate Factor value is just that; a recommendation. It is up to the developer to decide if they are truly taking advantage of the index, or hurting their system's performance.

1.22.10 Smart Indexing

The single most important factor in determining the effectiveness of an index is the uniqueness of index values. A unique index value is found faster than a value with multiple index occurrences. You can have multiple occurrences of index values if you have more than one row with the same data value in a column, or if you have a computed column (such as IHASH) whose values share the same result, or if the index is hashed.

Note that we are talking about unique index values, not data values. An index value may or may not be the same as a data value. Long text columns that are hashed when indexed have a higher probability of unique data values creating non-unique index values. An index loses its effectiveness as duplicate values increase and R:BASE must make more reads and comparisons.

Indexed columns also affect performance when adding or changing data in a table. Indexed columns must be updated when a row is added or when the index column value is changed. The number of indexed columns in a table affects the speed with which rows are added or changed in the table. Take this into account when defining indexes and constraints. It is faster to load and change data on a table with one indexed column than on a table with seven indexed columns.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.22.11 Summary

Using indexes effectively requires understanding how they work. This means deciding which columns to index, which data types to use, and whether to use IHASH. You also need to know how to improve performance when using a text index.

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

- Columns that are neither primary nor foreign keys, but are frequently referred to in queries and sorts
- Columns that have rules applied to them
- Linking columns in views
- The index is more efficient if each row contains a value
- The index is more efficient based on the uniqueness of the data

When properly implemented, indexes can greatly improve the data retrieval performance of your applications!

1.23 Information Management with R:BASE

R:BASE is a relational database product that lets you design sets of tables to store and retrieve your data easily. Each table contains information arranged in columns and rows about a single object or event, such as a customer list or a list of sales transactions. A column is a specific fact, such as a customer's name, about the object or event; a row is a cross-section of columns that is unique for a particular instance in the table.

An R:BASE database is a collection of tables. For each column in a table, you specify a data type that tells R:BASE what kind of data the column will hold - such as dates, currency, or text. You can also have columns that hold a value computed from other columns. For example, if you have sales stored in one column and a tax rate in another, a computed column can hold the tax on each sale (sales * tax rate).

The power of R:BASE lies in the control it gives you over the tables you create. You can manipulate the data from one or any combination of the tables in your database, and you can organize tables so that you rarely need to enter a particular piece of information more than once.

With R:BASE you can combine information from the tables in a database to provide answers for your questions and create another permanent table that stores the combined information. You can also create views, which are temporary tables that display all or part of the current contents of one or more tables. A view does not require you to store the data in more than one table -it always shows you the most up-to-date information, and it can combine data from several tables. A view is like a television screen that is split to show two camera angles at once.

1.24 International Characters

The R:BASE configuration file contains tables that define how R:BASE processes and prints characters. If you want to change how R:BASE evaluates characters, you can modify the information in these tables, which are described below.

R:BASE saves table configurations with the database. If you make modifications to these tables in the configuration file, use the PACK command with the WITH USER CASE option to compress the database and apply the new configuration.

The Case Folding Table

The Case Folding table establishes the correspondences between uppercase and lowercase characters, such as "A" and "a." R:BASE uses this table when testing characters for equality when the CASE setting is off.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Each line in this table starts with "CASEP" and is followed by two ASCII character codes corresponding to the uppercase and lowercase characters. For example, the following line shows that "a" (ASCII code 97) corresponds to "A" (ASCII code 65):

```
CASEP 97 65
```

The Collating Table

The Collating table equates two characters in sorting and inequality testing (>, >=, <, and <=).

Each line in this table begins with "COLLATE" and is followed by two ASCII character codes, whose corresponding characters are considered equal in a sorting sequence. For example, the following lines indicate that "a" (ASCII code 97), "ä" (ASCII code 228), and "A" (ASCII code 65) are all equal in a sorting order:

```
COLLATE 97 65

COLLATE 228 65
```

The Printer Table (DOS Only)

The Printer table tells the printer how to print certain characters. Some printers cannot print certain international characters, such as characters with accents or umlauts, so the printer must combine two or more characters to create the international character.

Each line in this table begins with "FOLD" and is followed by a character and its ASCII code, then one or more ASCII codes whose corresponding characters must be combined to create the first character. For example, the following line tells the printer how to print "à" (ASCII code 133); print "a" (ASCII code 97), backspace (BS), then print an accent "`" (ASCII code 96):

```
FOLD à 133 97 BS 96
```

If your printer can print a character without combining other characters, do not delete the line. Instead, edit the line. After "FOLD," enter the character, the character's ASCII code two times, then "00 00." For example, if your printer can print "à," edit the line as follows:

```
FOLD à 133 133 00 00
```

The Expansion Character Table

The Expansion Character table equates one character to two other characters. For example, you can equate "ß" to "SS." This table is used in tables, columns, variables, WHERE clauses, ORDER BY clauses, IF and WHILE commands, and indexed and non-indexed columns.

Each line in this table begins with "EXPAND" followed by three ASCII character codes. You can have up to seven lines in this table. For example, the following line equates "ö" (ASCII code 246) to "oe" (ASCII codes 111 and 101):

```
EXPAND 246 111 101
```

The Character Folding Table

The Character Folding table equates uppercase characters to lowercase characters. This table is used in string-manipulation functions.

Each line in this table begins with "LCFOLD" and is followed by two ASCII character codes. For example, the following line equates "A" (ASCII code 65) to "a" (ASCII code 97):

```
LCFOLD 65 97
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

The Case-Sensitive Collating Table

The Case-Sensitive Collating table lists characters and their position in the sequence order of all characters. This table is used when the CASE setting is on and when building indexes for columns with the TEXT data type.

Each line in this table begins with "COLLATEC" and is followed by an ASCII character code and its sequence position. For example, the following line indicates that "B" (ASCII code 66) is in the 76th position in the sequence order:

```
COLLATEC 66 76
```

If a character is not in this list, its sequence position number is the same as its ASCII character code.

1.25 Managing User Privileges

Security is vital to a database. Database administrators need to protect confidential and sensitive data and guard against accidental and inadvertent edits of data. R:BASE's optimistic concurrency control, table and row locking, constraints (primary and foreign keys), and data entry and delete rules provide a wide variety of data integrity controls. In addition, R:BASE has the following two options for controlling access to your data. The two options are:

- SQL Grant/Revoke system of access rights
- Form read and modify passwords

Each option provides a different type and level of security. Understanding and using these options enables application developers to provide appropriate security for any type of database and application.

1.25.1 Owner Identifier

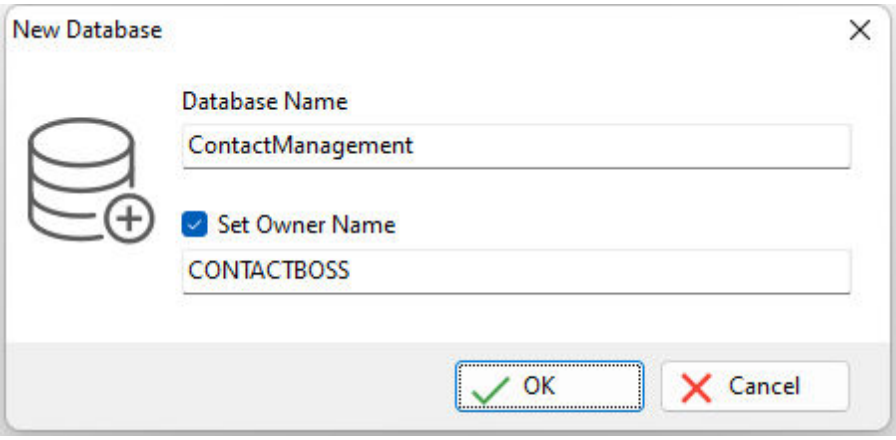
The first level of security is the owner identifier. Assigning an owner identifier keeps unauthorized users from connecting to the database. To use the R:BASE SQL Grant/Revoke control system of access rights, you must first assign an owner identifier to the database. The owner identifier for a database is a single password value.

Once you assign an owner identifier, you must enter that password to connect to the database. R:BASE automatically prompts for the password. The only other instance R:BASE will actually prompts for a password is when connecting a database and the current user (the value of the keyword USER) does not have access to any tables. If any permission is granted to PUBLIC, R:BASE no longer prompts automatically.

The owner password is stored in file 1 of the database files as an encrypted string. If you forget the owner password you cannot find it yourself. R:BASE Technologies has an internal utility that examines file 1 of a database and retrieves the owner password for the registered owner of the database. Keep your owner password secure.

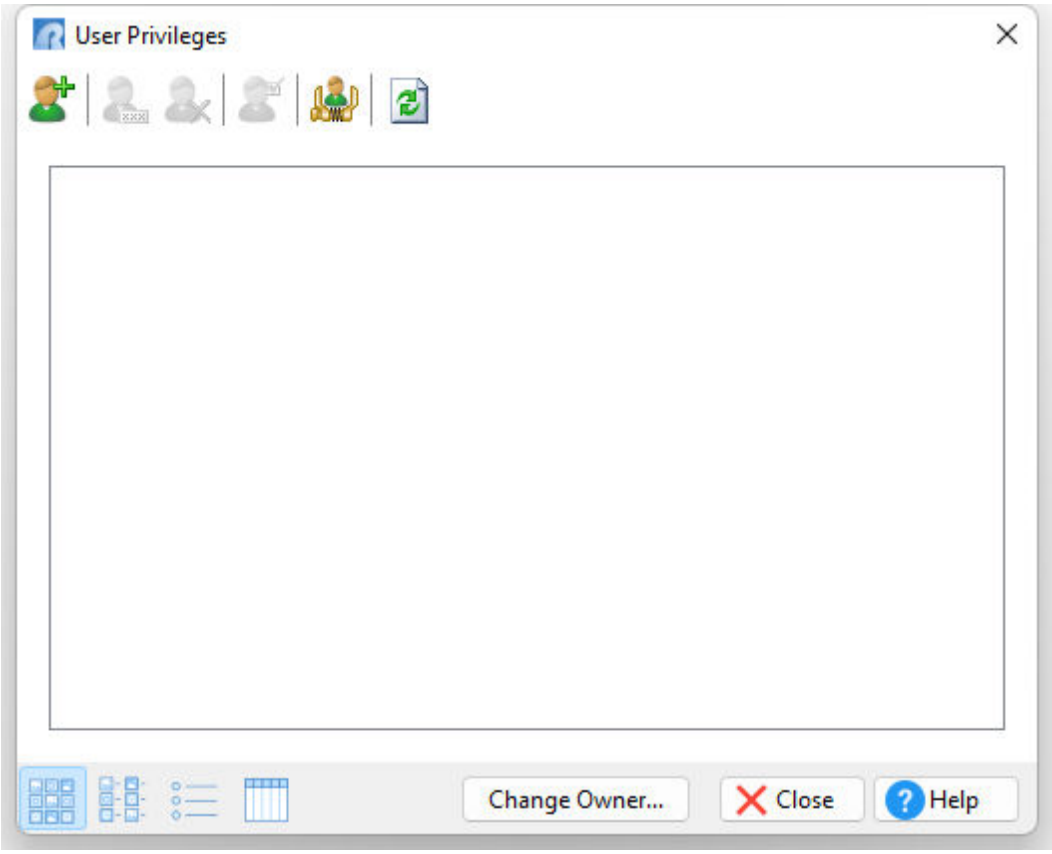
An owner identifier can be created when a database is first created.

Copyright © 1982-2024 R:BASE Technologies, Inc.



If an owner identifier was not specified at this time, the default value is set to NONE, which allows unrestricted access to the database.

To assign an owner identifier to a database, select "Utilities" > "User Privileges" from the main Menu Bar. Within this window, select the "Change Owner" button.



The owner identifier can also be changed, or initially assigned by using the RENAME command at the R> Prompt.

```
RENAME OWNER <oldownername> TO <newownername>
RENAME OWNER NONE TO <newownername>
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

With the owner identifier established, the value will be set as the current user for the R:BASE session. The owner name can be verified by entering SHOW USER at the R> Prompt.

After assigning the owner, R:BASE will prompt for the password the next time an R:BASE session connects to the database, or if the R:BASE USER account changes for the current session.

The owner can be specified at the R> Prompt using the SET USER or CONNECT commands:

```
SET USER ownername

CONNECT dbspec IDENTIFIED BY ownername
```

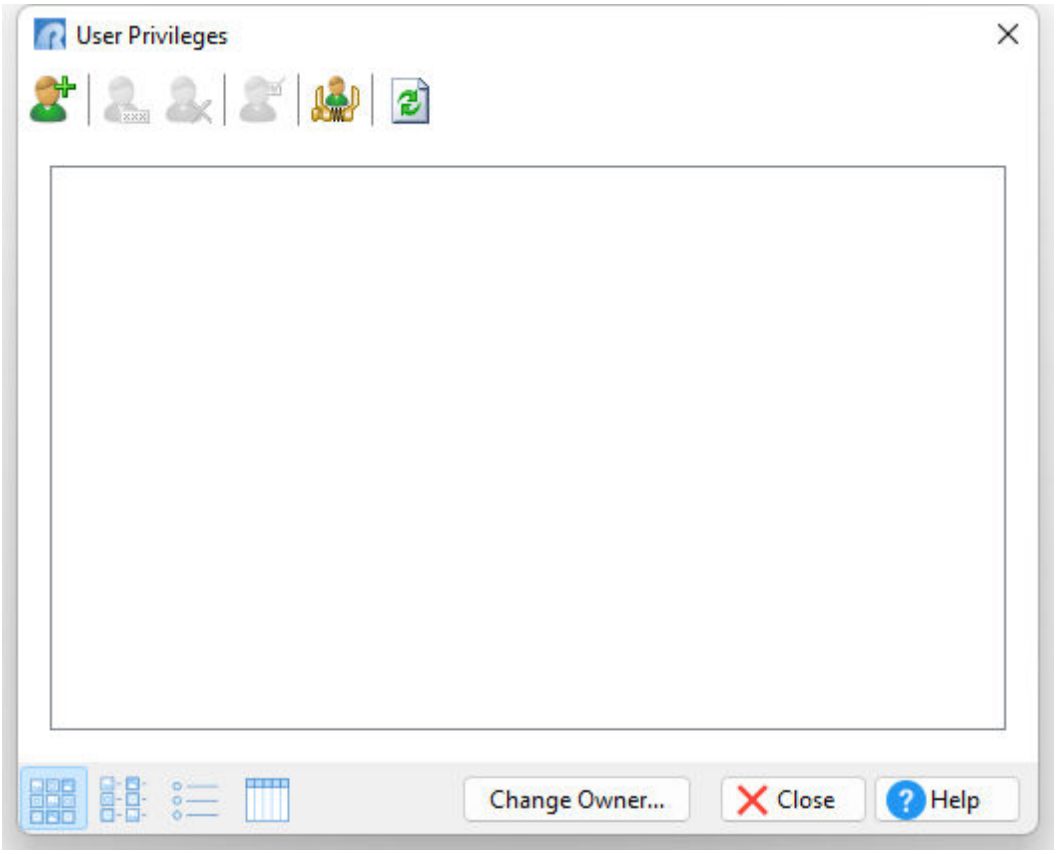
Only the owner of the database can use commands or menu options that modify database structure. Only the owner can assign access rights to other users. The owner of a database has all rights to all tables.

As the owner of the database, you decide the access rights you want to give to other users of the database. The SQL Grant/Revoke access rights are assigned by selecting "Utilities" > "User Privileges" from the main Menu Bar, or by using the GRANT and REVOKE commands at the R> Prompt.

1.25.2 Creating User Identifiers

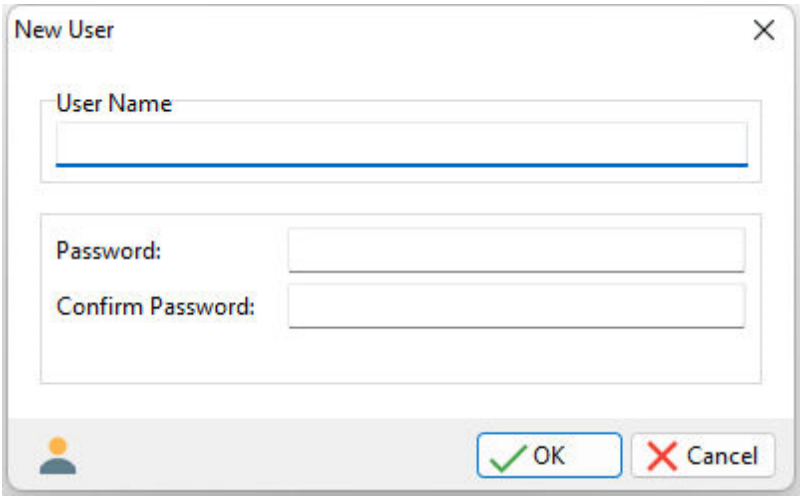
A user identifier is an assigned name by which a user can access a database. Adding user identifiers, or users, to the database can only be performed by the database owner.

To add a user, the owner can user the R> Prompt or the "User Privileges" interface. To use the graphic interface, select "Utilities" > "User Privileges" from the main Menu Bar. The User Privileges interface allows database administrators to easily create a list of users with various access rights to database tables. A list of users may or may not already be listed.

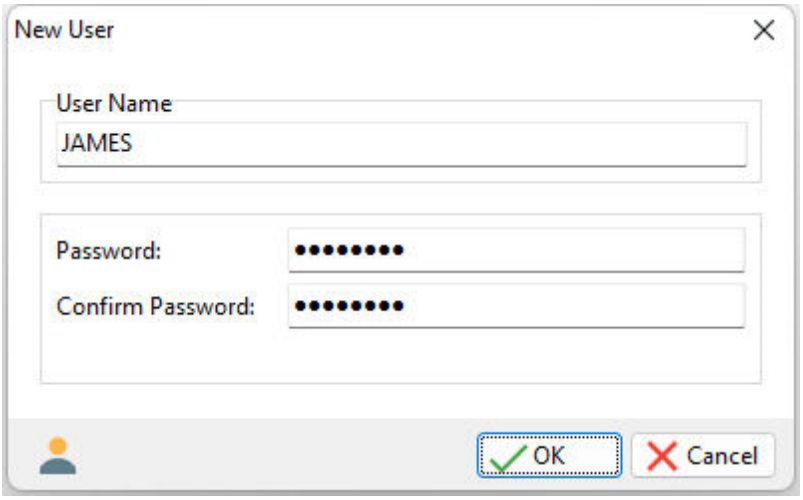


Copyright © 1982-2024 R:BASE Technologies, Inc.

Next, select the "New User" button. The following dialog will be displayed to enter the user name and password. If the database is to take advantage of Integrated Windows Authentication, the User Name and Password must match that of the network user name and password.

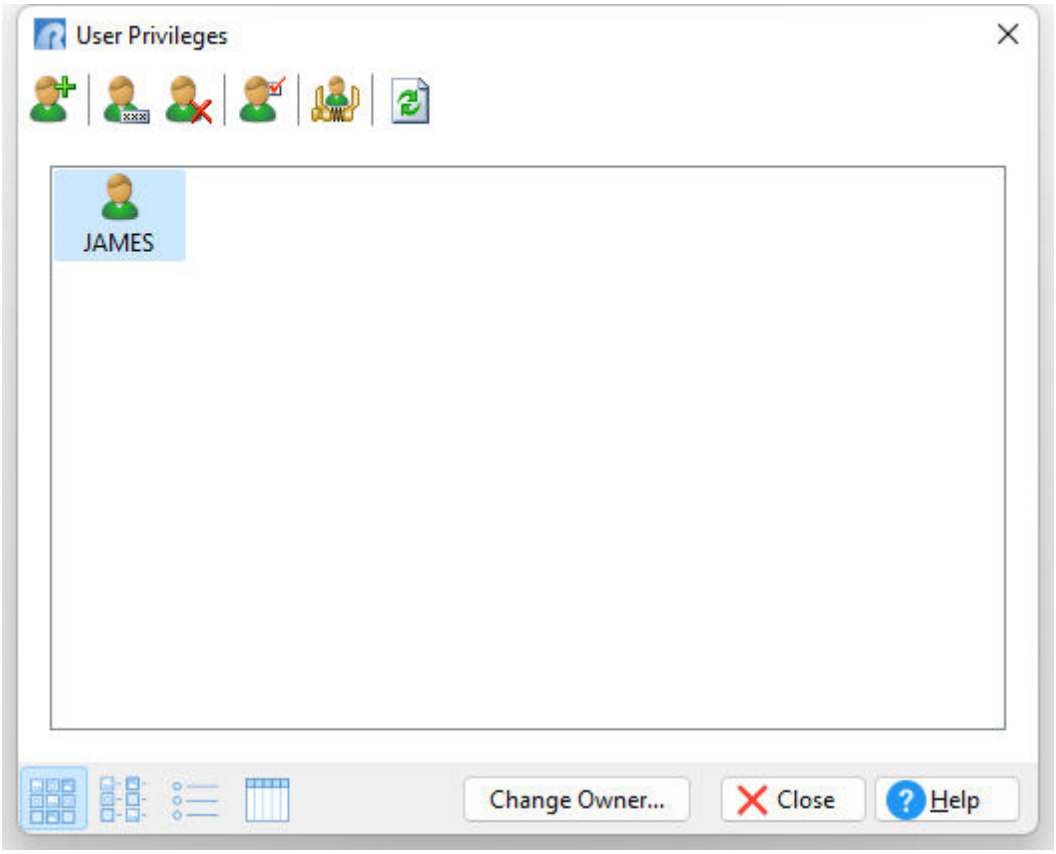


User identifiers have a maximum length of 36 characters. Passwords have a minimum length of three characters and maximum length of 36 characters.



After selecting the OK button, the new user will be added to the list of users for the database.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Other buttons exist to change the password for a user, delete the user, edit the user privileges to database tables, and to refresh the user list.

It is also possible for the database owner to add a user from the R> Prompt with the SET USER command syntax.

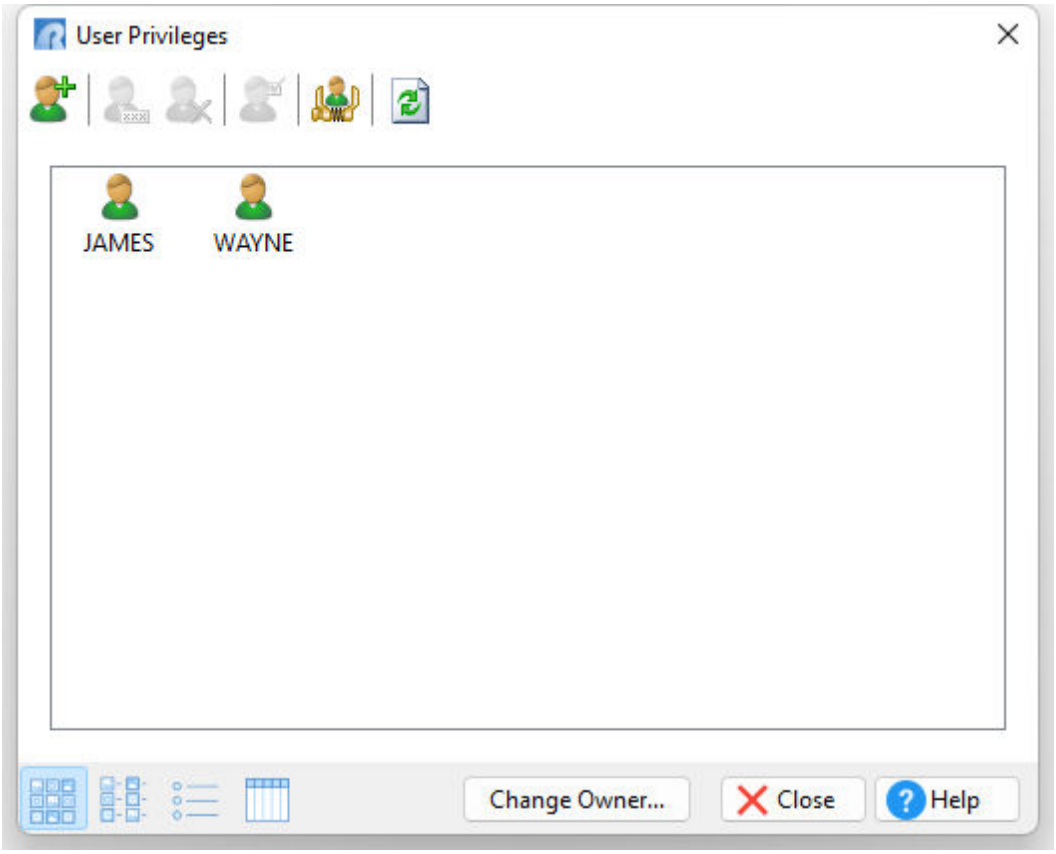
```
SET USER PASSWORD FOR <Userid> TO <Password>
```

The following command syntax will create the user "Wayne" with the password "wayne1998".

```
SET USER PASSWORD FOR WAYNE TO WAYNE1998
```

When you review the User Privileges window, the new user will be listed.

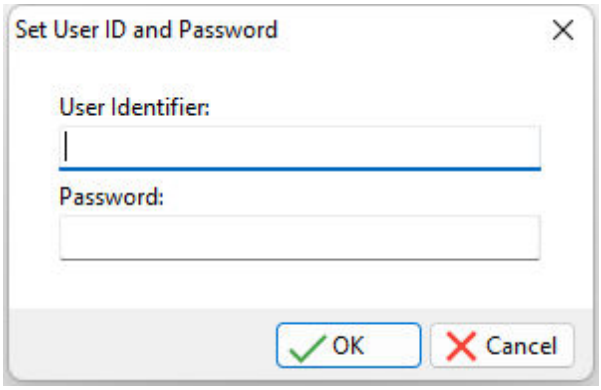
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.25.3 Setting User Identifiers

R:BASE might prompt for a user identifier when a database is connected. This happens when all the tables are protected and no access rights have been granted to PUBLIC. After you have assigned an owner identifier to the database, but before any access rights are granted to users, R:BASE prompts for a user identifier when the database is connected. If rights are granted to PUBLIC, no user identifier is needed to connect to the database and R:BASE does not prompt for one. If an access right has been granted to PUBLIC, you must specifically set the user identifier; R:BASE does not prompt you for one.

To enter or change the user identifier, select "Utilities" > "Set User ID and Password" from the main Menu Bar. R:BASE displays a dialog box and prompts you for your user identifier and password.

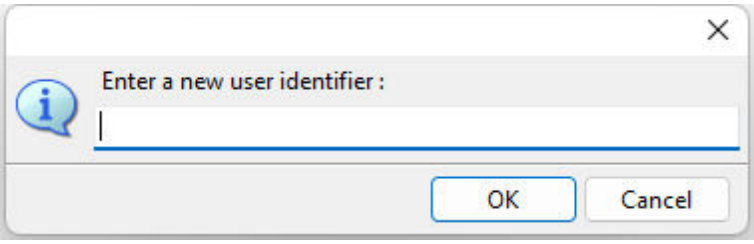


Copyright © 1982-2024 R:BASE Technologies, Inc.

When you enter your user identifier and optional password, R:BASE puts the value you enter into the keyword USER. The user identifier can be the owner identifier, the name of a user who has been granted access rights to some of the tables in the database, or PUBLIC.

Or, to enter or change the user identifier, use the SET USER command at the R> Prompt.

```
SET USER
```



You can also enter the following command at the R> Prompt to run R:BASE with your user identifier:

```
SET USER <Userid>
```

User identifiers have a maximum length of 36 characters. Passwords have a minimum length of three characters and maximum length of 36 characters.

Only one user identifier is currently active. The SHOW USER command displays the current user identifier.

```
SHOW USER
```

To capture the current user identifier in a variable, use the command:

```
SET VAR vUser = (CVAL('USER'))
```

When a user enters a user identifier or password in a dialog box, the user identifier is not displayed on screen. The keyword USER is not the same as the network ID or NAME.

If a user forgets their user identifier, the database owner can tell that user what the user identifier is, or reassign it.

R:BASE does not check to see that a valid user identifier was entered in response to the SET USER command or other prompt. The user identifier is checked at the time a command is executed. R:BASE checks the current value of the keyword USER to see if that user identifier has the appropriate permission to execute the requested action. For example, the SELECT command requires SELECT permission on the table(s) referenced in the SELECT command. When the command is executed, R:BASE checks to see if the current user identifier has SELECT privileges on those tables.

When a specified user is provided with access to a table in the database, that user does not have the ability to use the Designers for forms, reports, and labels.

1.25.4 User Passwords

The R:BASE password system is a simple process providing control over access to the data in your database on a table-by-table level. To gain access to the database, the correct user identifier and corresponding password must be entered.

R:BASE may prompt for a password when a database is connected. This happens only if all tables are password protected and no rights have been granted to PUBLIC. If any permission is granted to PUBLIC, R:BASE no longer prompts automatically.

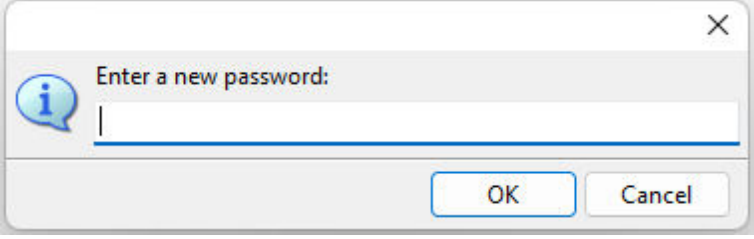
Copyright © 1982-2024 R:BASE Technologies, Inc.

Only one password can be currently active in an R:BASE session. Passwords are specific to user identifiers. The database owner cannot look at users' passwords within R:BASE, but the owner can reset the password for a user identifier. The owner can also reset the password for a user identifier.

To add or change a password, connect to the database with the user identifier and enter the following command line:

```
SET USER PASSWORD
```

If no password is set for the user identifier, R:BASE will prompt for a new password.



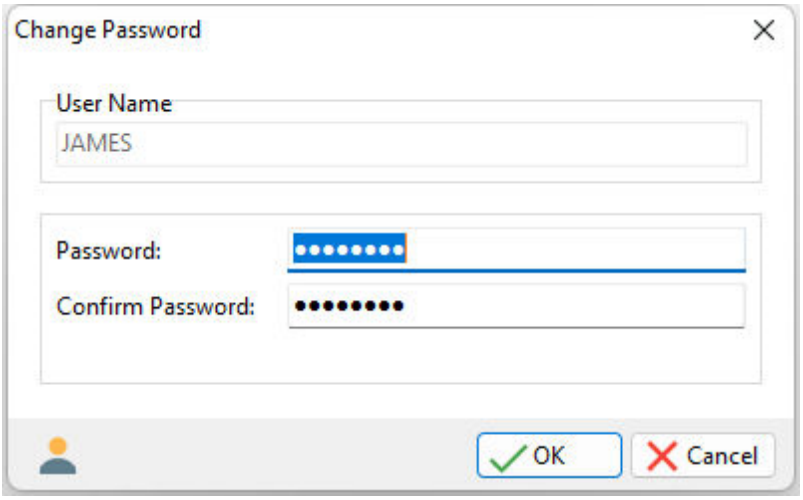
If a password already exists for the user identifier, R:BASE first prompts for the user identifier's existing password for confirmation.

After the existing password is confirmed, R:BASE will then prompt for the user identifier's new password. A user can cancel a password by entering NONE.

If the database owner is the current user, the database owner can assign him/herself a password using the SET USER PASSWORD command; however, if the database owner forgets the assigned password, the password cannot be found or changed. R:BASE Technologies has an internal utility that retrieves the owner password for the registered owner of the database.

As the database owner, a user's password can be changed for the connected database. The owner can user the R> Prompt or the "User Privileges" interface to do so.

For the graphic interface, select "Utilities" > "User Privileges" from the main Menu Bar. In the interface, select a specific user and then select the "Change User Password" button. The password can be changed by editing the password fields. The fields can also be left blank to delete a password.



To change a user's password at the R> Prompt, enter the following command line:

```
SET USER PASSWORD FOR <Userid> TO <Password>
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

Enter the current password when R:BASE prompts you for it, then when R:BASE prompts you for a new password, enter the new password, or NONE to leave the password empty.

R:BASE does NOT check the validity of the password when it is entered in response to the SET USER or other prompt. It is checked at the time a command is executed. R:BASE checks the current value of the keyword USER to see if that user has the appropriate permission to execute the requested action. For example, the SELECT command needs READ permission, so when the SELECT command is executed, R:BASE checks to see if the current user (password) has read/select permission on the tables.

1.25.5 Using GRANT/REVOKE Access Rights

The SQL Grant/Revoke access rights system assigns, removes, and lists access rights to the database. Access rights determine which actions a user can and cannot perform on the database. The owner of the database can assign and remove access rights to anyone; however, users can't assign or remove access rights unless the owner has given them permission.

The SQL Grant/Revoke system is in effect as soon as you assign an owner identifier to the database. Once an owner identifier is assigned, only the owner can access data until the owner grants permissions to other users. Once you assign an owner identifier, you must assign permissions to users on specific tables for those users to access data in the database. If permissions are not explicitly granted to users, they cannot access any data in any table. Each user can have a different set of user privileges for the same table, and you can grant a user the right to grant user privileges to others. Permissions can be granted to all users by specifying PUBLIC as the user identifier.

The database owner can grant the following levels of permissions:

SELECT	Permission to read data from the specified table. Generally, if you assign INSERT, DELETE, or UPDATE rights to a user, you also need to assign them SELECT rights. If SELECT permission is not assigned, the user cannot view data from the table.
INSERT	Permission to add new rows to the specified table. To add data with a form or the Data Browser, the user must be assigned INSERT permission. In addition, importing data or using the LOAD or INSERT commands from the R> prompt requires INSERT permission.
DELETE	Permission to delete rows from the specified table. For users to delete a row of data or all rows from a table, they must be granted DELETE permission.
UPDATE	Permission to modify data in the specified table on a column by column basis. UPDATE permission can be granted without a column list specified, giving the user permission to change the data in any column in the table. When granted with a column list, UPDATE permission lets the user change data in the specified columns only.
ALTER	Permission to modify the structure of the specified table. Normally, only the owner of a database can change the structure of a database, such as adding new columns to a table, changing the definition of existing columns, or dropping tables and views. The owner can grant permission to a user to change the column definitions in a table by assigning the user ALTER permission. In addition to changing the structure of a database by modifying table definitions, the ALTER permission gives the user permission to DROP the specified table or view.
CREATE	Permission to create new, permanent tables in the database (STATICDB OFF only). When STATICDB is set to on, any user can create a new table or view, that table or view is temporary, however, and automatically removed when the database is disconnected. Additionally, any user can create a permanent view or a table using one of the relational commands PROJECT, INTERSECT, UNION, JOIN without having CREATE permission assigned. The CREATE permission lets the owner give another user the right to add tables to the database using the CREATE TABLE command.
REFERENCES	Permission to create a table with a foreign key that references a table with a primary key.
WITH GRANT OPTION	Permission to grant the specified permission to other users. This permission is used to allow other users to grant access rights. You might want to assign a department supervisor, for example, the WITH GRANT OPTION so that they can assign access rights to users in their department.

Copyright © 1982-2024 R:BASE Technologies, Inc.

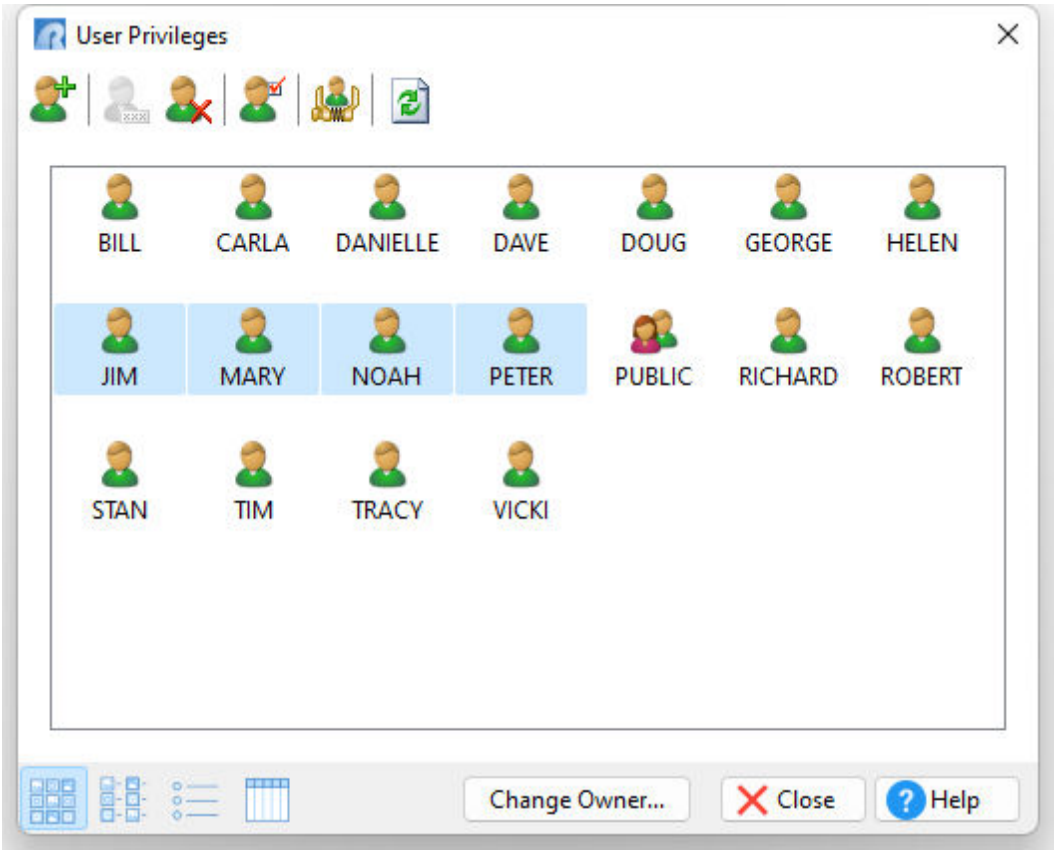
When using forms, R:BASE looks for Grant/Revoke permissions on the form tables. In general, to add data using a form, INSERT permission is required. To edit data, UPDATE permission is required. To delete rows, DELETE permission is needed. Form lookup expressions require SELECT permission on the lookup table.

Reports and labels require SELECT permission on the driving table or view, as well as SELECT permission on the lookup table(s) used in any expressions.

A user identifier can be granted one or more of these permissions for any table. The owner can grant the same permission to many different users. To grant or revoke access rights to users, the database owner can use the User Privileges interface or commands at the R> Prompt.

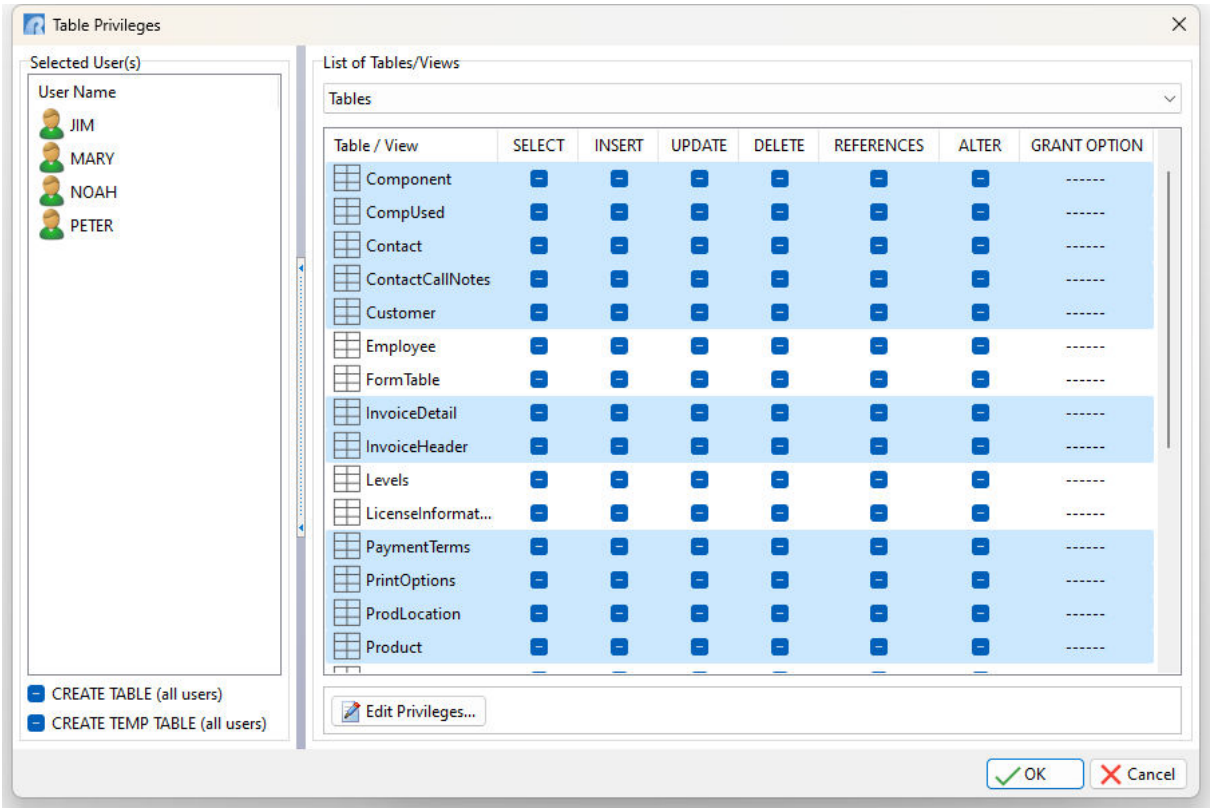
1.25.5.1 User Interface

The owner can also grant and revoke permissions to users using the User Interface. To open it, select "Utilities" > "User Privileges" from the main Menu Bar. In the interface, select one or more specific users and then select the "User Privileges" button.



The Table Privileges dialog window will be displayed where the selected users can be assigned table access rights for the entire database. The table permissions to SELECT, INSERT, UPDATE, DELETE, REFERENCE, and ALTER can granted and revoked for the tables. Options are also available to grant "CREATE TABLE" permissions (permanent and/or temporary table) to the user(s). Users who have been granted this permission have all privileges on the tables they create, including the WITH GRANT OPTION. However, users do not have privileges on any other tables in the database unless they are specifically granted permission by the owner.

Copyright © 1982-2024 R:BASE Technologies, Inc.

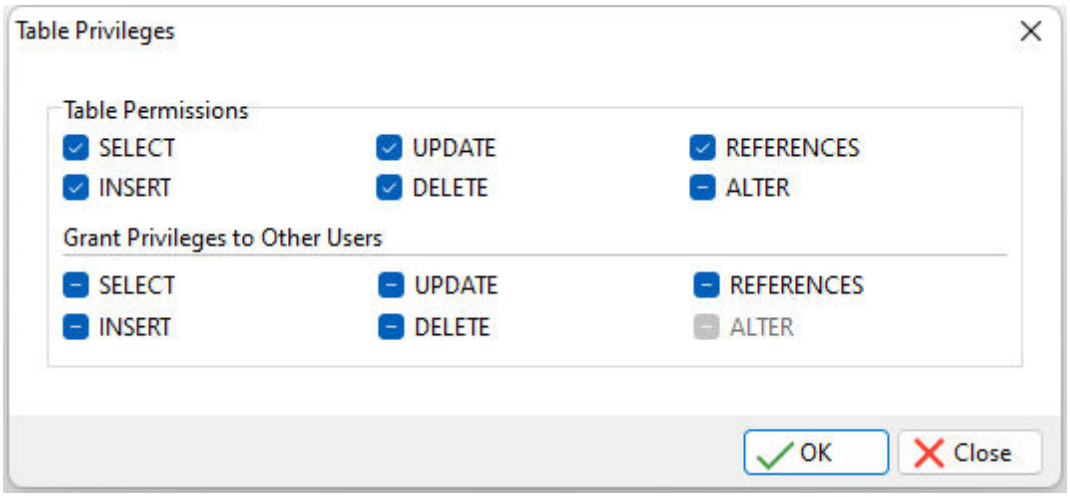


Using the mouse with the [Shift] and [Ctrl] keys, a great deal of flexibility is available to provide table privileges, such as:

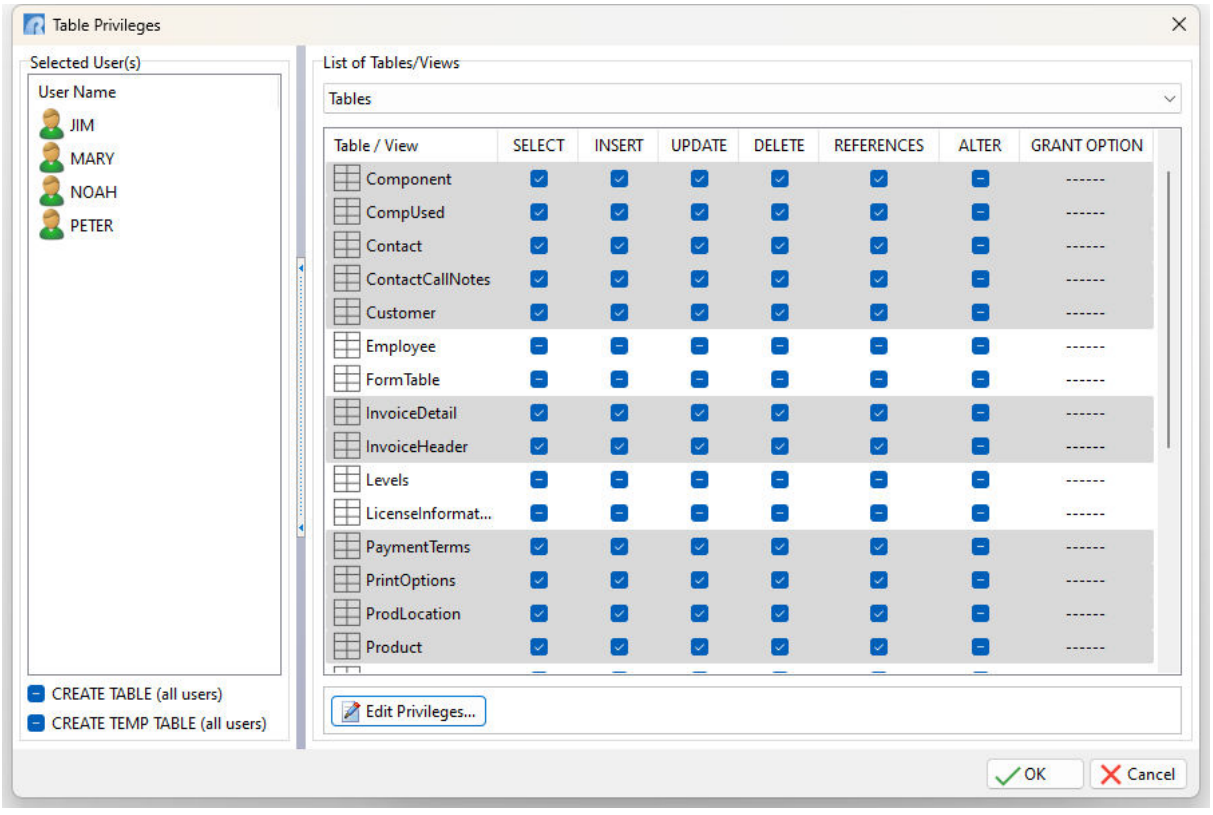
- Press and hold the [Ctrl] key and left click the mouse to select multiple tables/views individually
- Press and hold the [Shift] key plus the [Up/Down Arrow] keys to select multiple tables/views consecutively
- Select the [Ctrl] key plus [A] to select all the tables/views

After selecting the appropriate tables for the users, select the "Edit Privileges" button to assign the rights to the selected tables. The following dialog will be displayed to individually enable/disable the specific permission.

Copyright © 1982-2024 R:BASE Technologies, Inc.

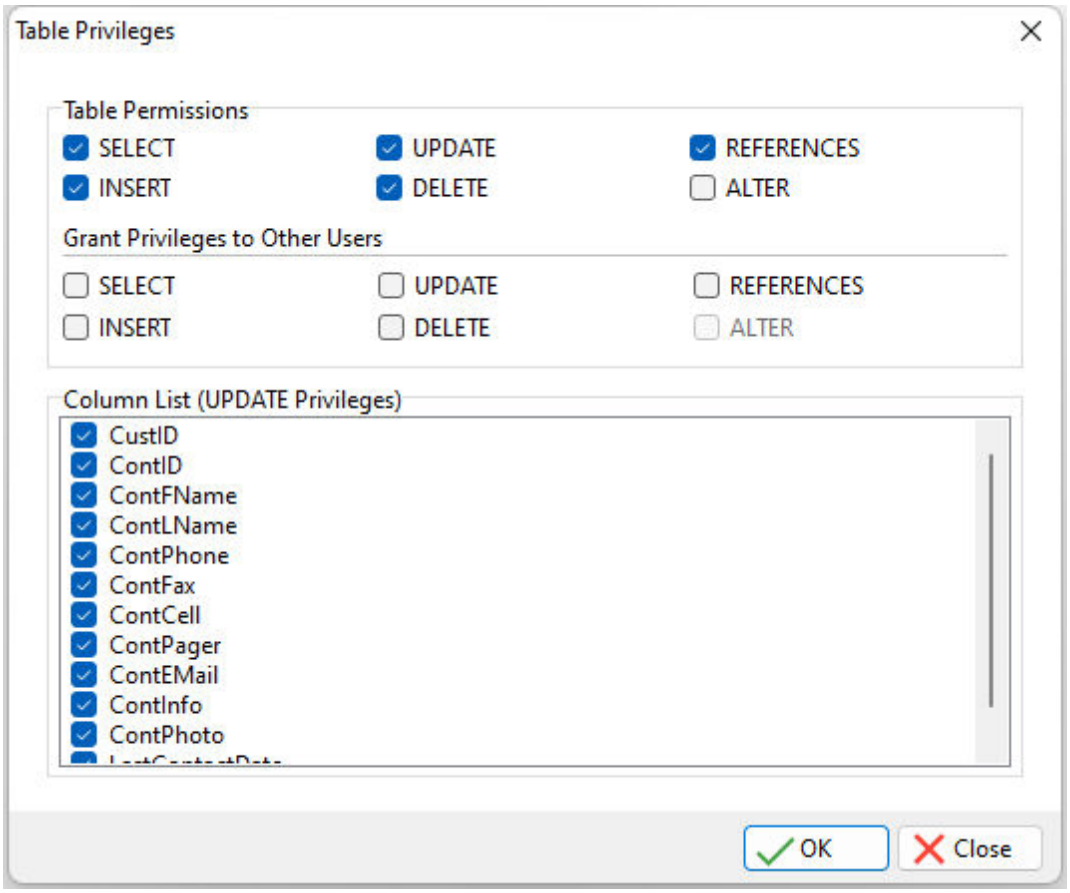


The changes will be saved.

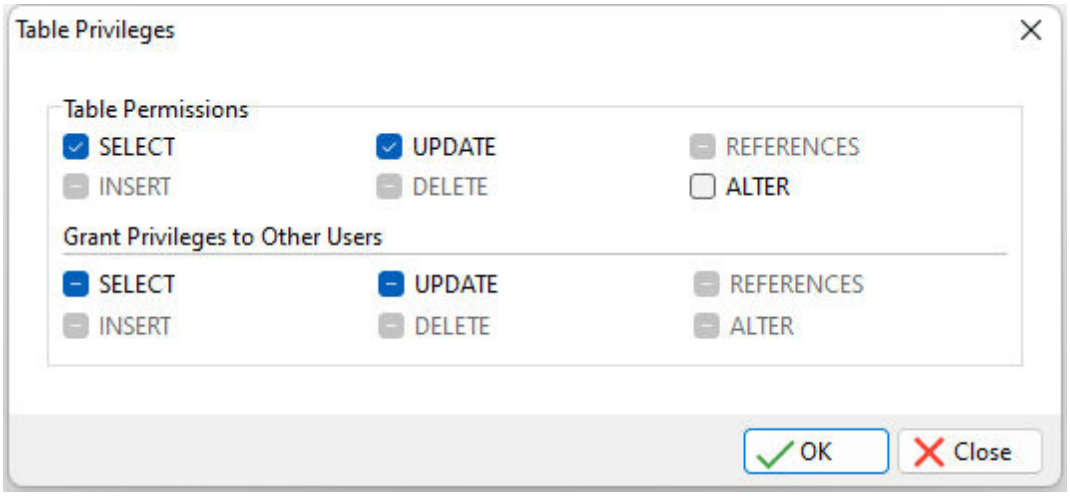


When privileges are viewed for an individual table, column specific privileges can be assigned to the UPDATE permissions. In the dialog below, this would permit a list of user to update some columns and not others.

Copyright © 1982-2024 R:BASE Technologies, Inc.

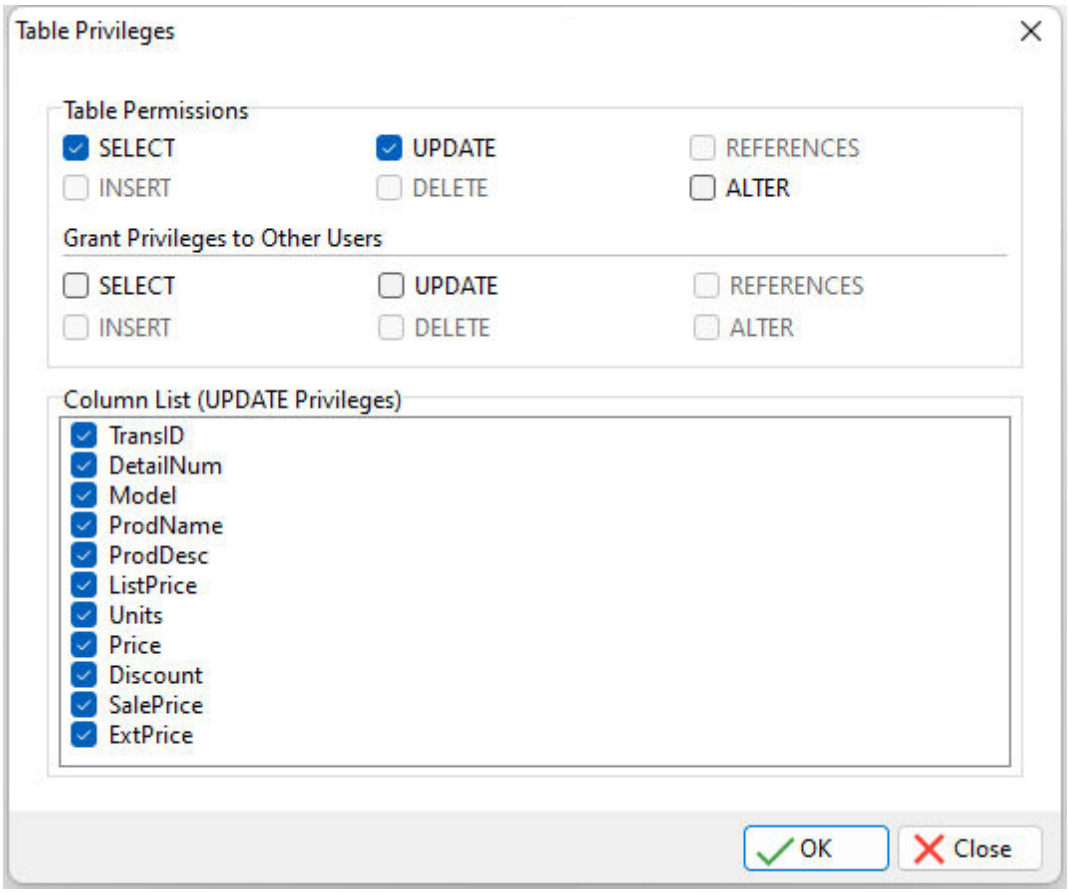


When privileges are being set for views, the INSERT, DELETE, and REFERENCES options are disabled. The same options are disabled when browsing permissions if both tables and views are selected. For this reason, it is easier to provide permissions for tables and view separately by using the drop down box within the "List of Tables/Views" panel.



When privileges are considered for an individual view, column specific privileges can be assigned to the UPDATE permissions.

Copyright © 1982-2024 R:BASE Technologies, Inc.



When permissions need revoked for any user, the check boxes would simply need to be unchecked.

If an owner revokes all of a user's privileges from the database, the user's password is automatically also revoked.

1.25.5.2 Using Commands

The owner can also grant and revoke permissions to users using the GRANT and REVOKE commands at the R> Prompt or within a command file. Permissions are granted on a table by table basis.

To grant display, edit, and enter permissions to the user Doug, the following commands would be used:

```
GRANT SELECT ON Employee TO DOUG
GRANT UPDATE ON Employee TO DOUG
GRANT INSERT ON Employee TO DOUG
```

To grant edit permission for specific columns in the Contact table to the user Bill, the following command would be used:

```
GRANT UPDATE ( +
ContFName  +
ContLName  +
ContPhone  +
ContFax  +
ContCell  +
ContPager  +
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
ContEMail  +
) ON Contact TO BILL
```

The same permission on a table can be granted to many users in one command, but permissions on each table must be granted separately. You cannot grant permissions to many tables in one command. To grant display, edit, and enter permission in the Customer table to several users, the following command would be used:

```
GRANT SELECT, UPDATE, INSERT ON Customer TO GEORGE, RICHARD, MARY,
HELEN
```

There is also the GRANT command parameter, ALL PRIVILEGES, for the owner to grant display, edit, enter, and delete permissions at one time. The ALTER and CREATE permissions must be granted individually.

```
GRANT ALL PRIVILEGES ON InvoiceHeader TO DOUG
GRANT ALL PRIVILEGES ON InvoiceDetail TO DOUG
```

The following command grants the user John, who is not the database owner, permission to alter table Customer.

```
GRANT ALTER ON Customer TO John
```

The following command line grants the user John, who is not the database owner, permission to create tables.

```
GRANT CREATE TO John
```

As the owner of a database, user privileges can be revoked from users. If the database owner or other users have assigned user privileges with the WITH GRANT OPTION, a user can revoke only the user privileges that the user granted to others.

The following command revokes permission granted to Jane to display or print data, or add rows to the InvoiceHeadertable.

```
REVOKE SELECT, INSERT ON InvoiceHeader FROM JANE
```

The following command revokes the UPDATE user privilege granted to Sam for all tables and views in the database.

```
REVOKE UPDATE FROM SAM
```

The following command revokes all user privileges granted to Sam, except those granted to him as a member of PUBLIC.

```
REVOKE ALL PRIVILEGES FROM SAM
```

The following command revokes all user privileges for all tables and views for all users.

```
REVOKE ALL PRIVILEGES FROM SAM, JANE, JOHN, PUBLIC
```

When a user creates a new table with one of the relational commands or by choosing "Save Result As" in the Data Browser, the user is automatically assigned all access rights except for the WITH GRANT OPTION to the new table. When a user with CREATE permission makes a new table, that user is automatically assigned all rights to the table, including the WITH GRANT OPTION.

The LIST ACCESS command shows the permission as ALTER when access rights are automatically assigned to new tables. The ALTER permission automatically implies all other privileges.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.25.6 Displaying Permissions

The owner of the database can list all user identifiers and assigned permissions by using the LIST ACCESS command. The owner can list access rights for a particular user, for a permission type, for a table, or for all permissions. A user can list their permissions only. Some of the LIST ACCESS options an owner can use include:

To display all access rights granted for the entire database, use:

```
LIST ACCESS
```

To display all access rights granted to all tables for the specified user, use:

```
LIST ACCESS FOR <username>
```

To display all access rights granted to all users for the specified table, use:

```
LIST ACCESS ON <tablename>
```

To display all access rights granted for all users and all tables with the specified permission, use:

```
LIST ACCESS permission
```

1.25.7 Form Passwords

Forms can be protected with password at two instances, before a form is launched in the designer and before a form is launched a runtime.

The "Design-Time" Password allows the ability to set a password to protect users from accessing the form in the Form Designer. Once the design time password is set, the password will be required the next time the form is opened in the Form Designer. This password is above any table level permissions for tables used in the forms.

The "Run-Time" Password allows the ability to set a runtime password to protect users from accessing the form and viewing the data. When the form is launched in enter, edit or browse mode, the end-user will be required to enter the runtime password. This password is above any table level permissions for tables used in the forms.

1.25.8 Password Maintenance

When adding, editing and deleting users while managing different permissions for users, it is always best to perform regular maintenance on the R:BASE SYS_PASSWORDS system table, which stores the password information.

To perform the maintenance, use the PACK PASSWORD command at the R> Prompt or in a command file.

```
PACK PASSWORD
```

The command will clean out bogus rows from the system table. This command can be used in a multi-user environment.

All user permissions can be unloaded from the database into a file with the UNLOAD command. The resulting output will only contain the existing GRANT permissions. No user names or password are provided.

```
OUTPUT UserPriv.str
UNLOAD STRUCTURE FOR ACCESS
OUTPUT SCREEN
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

All database users, passwords, and user permissions can be unloaded from the database into a file by unloading the entire database structure into a file.

```
OUTPUT dbname.str
UNLOAD STRUCTURE
OUTPUT SCREEN
```

The resulting output file will contain the complete database schema, tables, views, constraints, rules, etc, but the specific SET USER and GRANT commands that result in the database user privileges can be reviewed.

Note: The UNLOAD STRUCTURE command will also unload the database OWNER name. The generated file must be kept secure.

1.25.9 Integrated Windows Authentication (IWA)

Integrated Windows Authentication (IWA) is the automatically authenticated connections between Microsoft IIS, Internet Explorer, and other Active Directory aware applications.

In R:BASE, support for Integrated Windows Authentication uses the security features of Windows clients and servers and allows access to an R:BASE database when the Windows user name and password matches a defined R:BASE database user name and password. The current Windows user information on the client computer is supplied to R:BASE and does not prompt user for additional user name and password.

In order to use this feature, R:BASE database administrators must create an owner and user privileges for the database files to take full advantage of IWA support.

In addition to creating the database user privileges, the operating condition WINAUTH must be set to ON when prior to users connecting to the database. WINAUTH can be set within the R:BASE configuration file, or defined within an application startup file.

Requirements:

- WINAUTH must be set ON before connecting to the database.
- Database must contain an owner and user privileges that match Windows network accounts.

Notes:

- R:BASE will accept user names that contain spaces.
- Passwords are case-sensitive.
- If the authentication exchange initially fails to identify the user, R:BASE will prompt the user for a user name and password.
- A new CVAL Function (CVAL('WINAUTH')) has been established to check if WINAUTH is on.
- When an R:BASE user successfully connects to the database via IWA, the current R:BASE USER identifier is set to the current Windows user name.

1.25.10 User Permission Functions

(CVAL('ISOWNER'))

Returns the value of YES when the current user has OWNER permissions, and NO when the user does not.

(TINFO(arg1,*arg2*,*arg3*))

Returns access right information for the current user.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Where: arg1 is zero (0), which specifies to show table permissions
arg2 is the system table ID (SYS_TABLE_ID) value (can be located in the SYS_TABLES system table)
arg3 can be:
a) A specific system column ID (SYS_COLUMN_ID) in that table
b) A value of 0 which means show permissions that apply to all the columns in that table
c) A value of -1 which means show permissions that apply to any of the columns in that table

Remarks:

- TINFO returns a comma delimited string of the access rights.
- In most instances table and column privileges are identical. Cases where they are not are autonumber column (no INSERT), computed columns, (no INSERT or UPDATE), and where the UPDATE privilege was granted to specific columns only.
- Permissions are returned based upon the current USER.

Examples:
The following example is based upon a database configured with an OWNER and users, where limited permissions are supplied to a user Jim for the Component table.

While connected to the database as the OWNER, the system table ID and system column ID can be found for the Component table and CompDesc column.

```
SELECT SYS_TABLE_ID FROM SYS_TABLES WHERE SYS_TABLE_NAME = 'Component'
 SYS_TABLE_ID
 ------------
           29

R>SELECT SYS_COLUMN_ID FROM SYS_COLUMNS WHERE SYS_COLUMN_NAME = 'CompDesc'
 SYS_COLUMN_ID
 -------------
           188

R>SET USER JIM JIM999
```

Example 01:

```
-- Permissions for the column CompDesc
R>SET VAR vUserInfoColumn = (TINFO(0,29,188))
R>SHOW VAR vUserInfoColumn
SELECT, UPDATE
```

Example 02:

```
-- Permissions for all column. Only SELECT is returned since that is the only permission applied to all columns.
R>SET VAR vUserInfoAllColumns = (TINFO(0,29,0))
R>SHOW VAR vUserInfoAllColumns
SELECT
```

Example 03:

```
-- Permissions for any column. Both SELECT and UPDATE are returned since there are some columns with both of those permissions.
R>SET VAR vUserInfoAnyColumn = (TINFO(0,29,-1))
R>SHOW VAR vUserInfoAnyColumn
SELECT, UPDATE
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.25.11 Generating Random Numbers

The basic procedure for generating a random number is to start with a "seed" number and then perform calculations on the "seed" number to generate the actual random numbers. Often the random number generated is used as the "seed" number for the next iteration. The thousandths part of the time format is used as the "seed" number in an R:BASE command file to generate random numbers. Various SuperMath functions and other arithmetic operations generate the random number. The calculations can be simple or complex; often the simpler calculations generate the more random number.

The following examples of calculations that generate a random number use the same "seed" number each time and also use the complete time value in the calculation. The time format is set so that it looks like a number.

```
SET TIME FORMAT HH:MM:SS.SSS
SET VAR vTime INTEGER = (INT(FORMAT(.#TIME,'HHMMSS')))
SET VAR vRandom1  = +
  (NINT(((IFRC(.#TIME))*100)/(NINT(CTXT((.vTime)))/10000)))
SET VAR vRandom2 = +
  (IFRC(.#TIME)+NINT(LOG(.vTime))+NINT(SQRT(.vTime)))
SET TIME FOR HH:MM:SS AP
```

1.25.12 Generating Random Text

A random string of numbers and letters can be used as a user identifier or password. Many database applications store actual GRANT/REVOKE access names in a look-up table. The user uses their own name to retrieve a particular granted user identifier and level of access from the look-up table. In this type of application, the user identifier is never known by any user of the database. To ensure security to the data, the user identifiers stored in the look-up table are changed on a regular basis. This command file uses the #TIME and #DATE system variables and various date and time functions to create a random character string of numbers and letters. The password look-up table is updated with the random string created by the program.

```
SET VAR vRandomText TEXT = NULL

-- The source variable, make sure there are
-- no spaces in the string.
SET VAR vKey TEXT = +
  ('A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,+
  U,V,W,X,Y,Z,0,1,2,3,4,5,6,7,8,9')

SET VAR vTime TIME = (.#TIME)
SET VAR vChar1 INTEGER = (ISEC(.vTime))

-- Require the first character to be a letter.
WHILE vChar1 > 26 THEN
  SET VAR vChar1 = (.vChar1 - 26)
ENDWHILE

-- Set character 2 to the minutes portion of the time.
-- This section needs a test for greater than 36,
-- if the value is out of range, the result is null
SET VAR vChar2 INTEGER = (IMIN(.vTime))

-- Set character 3 to the minutes divided by 6,
-- this will never be a number greater than 10.
SET VAR vChar3 INTEGER = (IMIN(.vTime)/6)

-- Set character 4 to the day portion of the date.
SET VAR vChar4 INTEGER = (IDAY(.#DATE))
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
-- Character 4 can never be greater than 36.
SET VAR vChar4 = (.vChar4 + 5)
IF vChar4 > 36 THEN
  SET VAR vChar4 = (vChar4 - 36)
ENDIF

-- Set character 5 to the number of the day
-- of the week, this is a number between 1 and 7,
-- and then multiply by the value of character 3.
-- The maximum value for this number is 70.
SET VAR vChar5 INTEGER = (IDWK(.#DATE))
SET VAR vChar5 = (.vChar5 * .vChar3)

-- Make sure this number is less than 36.
IF vChar5 > 36 THEN
  SET VAR ch5 = (.vChar5 - 36)
ENDIF

-- Set character 6 to the month portion of the date
-- and then add the value of character 5
-- (maximum for 5 is 34, plus 12 = 46).
SET VAR vChar6 INTEGER = (IMON(.#DATE))
SET VAR vChar6 = (.vChar6 + .vChar5)
IF vChar6 > 36 THEN
  SET VAR vChar6 = (.vChar6 - 36)
ENDIF

-- Set character 7 to the Julian date of the date,
-- then get the third and fourth characters from it.
-- This will be a number between 0 and 36.
-- Need to test for 0, when 0 this character is null.
SET VAR vChar7 INTEGER = (JDATE(.#DATE))
SET VAR vChar7 TEXT
SET VAR vChar7 = (SGET(.vChar7,2,3))
IF vChar7 = '00' THEN
  SET VAR vChar7 = (SGET(.vChar7,2,4))
ENDIF
SET VAR vChar7 INTEGER

-- The hour portion of the time plus 12, is a number
-- between 12 and 24.
SET VAR vChar8 INTEGER = ((IHR(.vTime))+12)

-- Add all the previous character values together
-- and divide by 8, the average of all the previous
-- characters
SET VAR vChar9 INTEGER = +
  ((.vChar1+.vChar2+.vChar3+.vChar4+.vChar5+.vChar6+.vChar7+.vChar8)/8)

-- Get the actual characters from the source variable.
SET VAR vRandomText = +
(SSUB(.vKey,.vChar1) + SSUB(.vKey,.vChar2) + +
SSUB(.vKey,.vChar3) + SSUB(.vKey,.vChar4) + +
SSUB(.vKey,.vChar5) + SSUB(.vKey,.vChar6) + +
SSUB(.vKey,.vChar7) + SSUB(.vKey,.vChar8) + +
SSUB(.vKey,.vChar9))

CLEAR VAR vChar%, vKey, vTime
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
RETURN
```

1.26 Modeless Windows (MDI)

R:BASE supports two types of windows: modal and modeless. Modeless, or non-modal, windows are used when the requested information is not essential to continue, and so the window can be left open while work continues elsewhere. Modeless windows are initiated in R:BASE using MDI (Multiple Document Interface) as a setting and command syntax parameter. Modeless forms let you shift the focus between the form and another form without having to close the initial form. The user can continue to work elsewhere in any application while the form is displayed. Modeless forms are harder to program, because users can access them in an unpredictable order. You have to keep the state of the application consistent no matter what the user does. Often, tool windows are shown in a modeless fashion.

The R:BASE "Find in Files" utility, accessible from the Utilities menu bar, is an example of a modeless dialog box.

A modal window must be closed before you can continue working with the rest of the application. Dialog boxes, launched using DIALOG or PAUSE 2 USING, are examples of windows that display important messages which are always modal.

1.26.1 MDI Setting

MDI is an available operating condition setting that controls whether R:BASE uses a modeless, or non-modal, window which can be left open while work continues elsewhere. MDI can be set ON or OFF, with the default being ON.

```
SET MDI ON
```

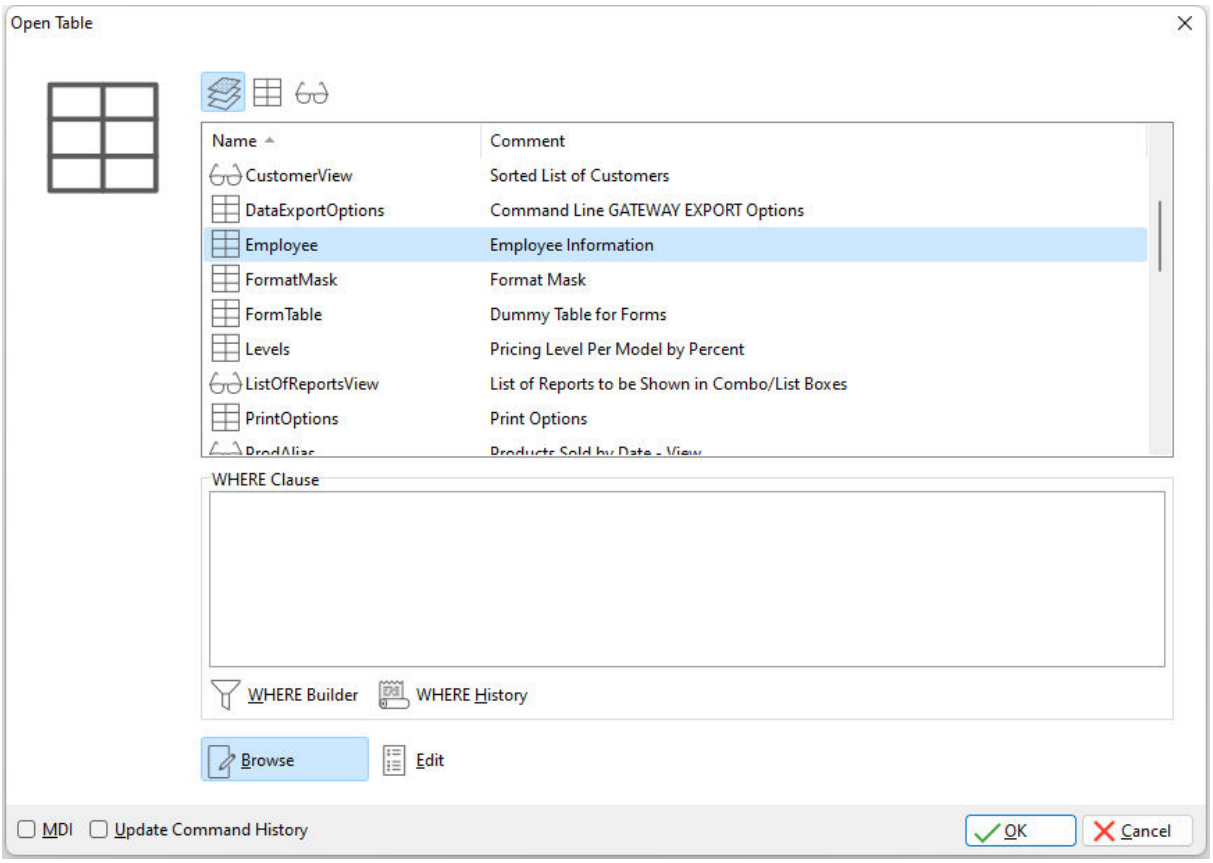
Users can adjust the setting to control the "MDI" option when running a form, printing a report/label to the screen, running an external form, or opening a table/view with the Database Explorer "Use Dialog to Open Table/View" setting, from the Database Explorer or toolbar. When the dialog window is opened, a check is placed in the "MDI" check box if MDI is ON or displays no check if MDI is OFF.

1.26.2 Displaying Data With MDI

The R:BASE Data Browser can also be launched to show table and view data in a modeless window. The Data Browser can be opened from the user interface with the Database Explorer and main toolbar, or by using the R:BASE command syntax at the R> Prompt and within a command file/application.

Using the main toolbar, and with a table/view selected, an initial dialog will launch with several options, including an "MDI" check box. If the MDI setting is set to ON, the check will be placed in the box when the dialog opens.

Copyright © 1982-2024 R:BASE Technologies, Inc.



When MDI is checked, all Data Browser windows launched in the user interface will be displayed as a modeless window. The window(s) will appear in the center of the screen and look just as if the window was launched without MDI, but the difference is you will be able to continue accessing menus in R:BASE without having to close the Data Browser. If the window is normalized and other aspects of R:BASE are used, the window will be located with other opened R:BASE modules. The focus can be returned to the Data Browser by selecting "Window" > "Data Browser..." from the main Menu Bar.

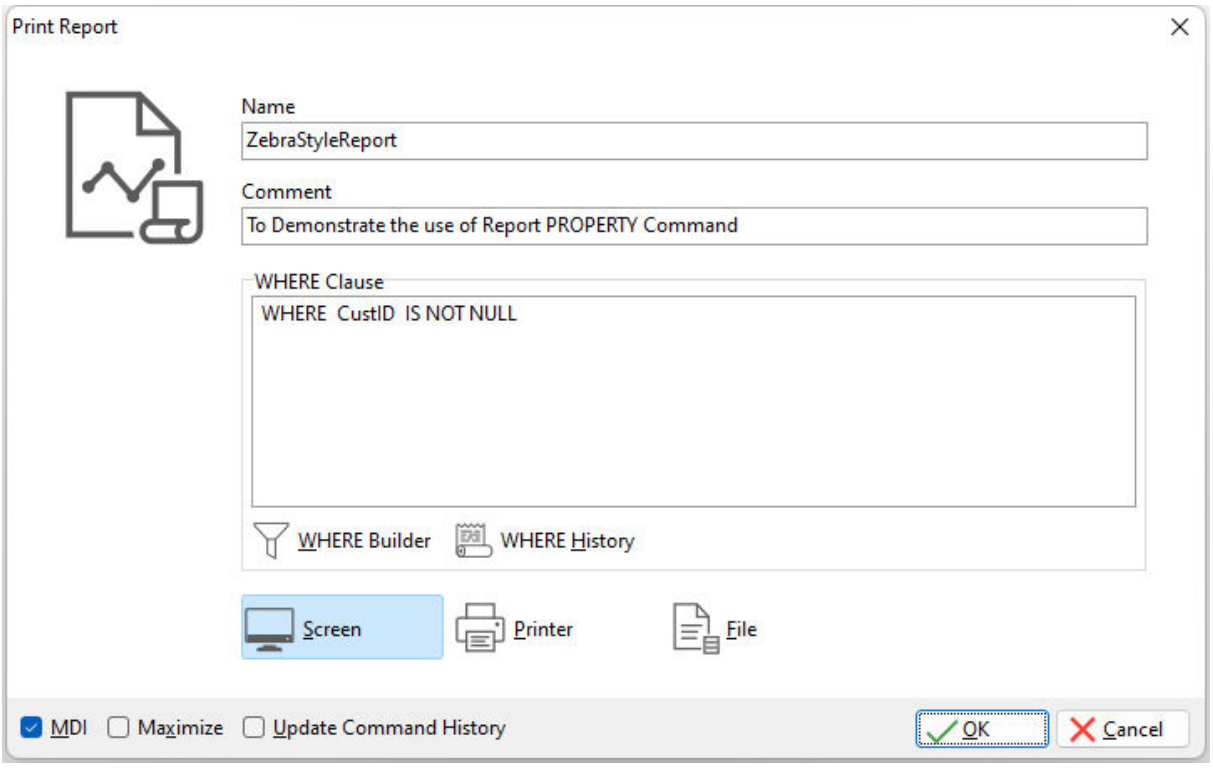
Using the R:BASE command syntax a modeless Data Browser window can be launched with the BROWSE and EDIT commands. Each command supports an MDI parameter to specify if the browser is to be opened as a modeless window.

1.26.3 Printing MDI Reports/Labels to the Screen

A report and label can be printed to the screen in R:BASE, where the preview screen can be displayed in a modeless window. The reports and labels can be opened from the user interface with the Database Explorer and main toolbar, or by using the R:BASE command syntax at the R> Prompt and within a command file/application.

With the user interface, an initial dialog will launch with several options including an "MDI" check box. If the MDI setting is set to ON, the check will be placed in the box when the dialog opens.

Copyright © 1982-2024 R:BASE Technologies, Inc.



When MDI is checked, the print preview will be displayed as a modeless window. The report/label will appear in the center of the screen and look just as if the window was launched without MDI, but the difference is you will be able to continue accessing menus in R:BASE without having to close the window. If you shrink and click outside of the window and use other aspects of R:BASE, the window will be located behind the R:BASE program window. The focus can be returned to the print preview by selecting "Window" > "MDI Windows" > "Reports" > "Report Name:" from the main Menu Bar. Or, the R:BASE program window can be normalized and moved aside to see the print preview window behind it.

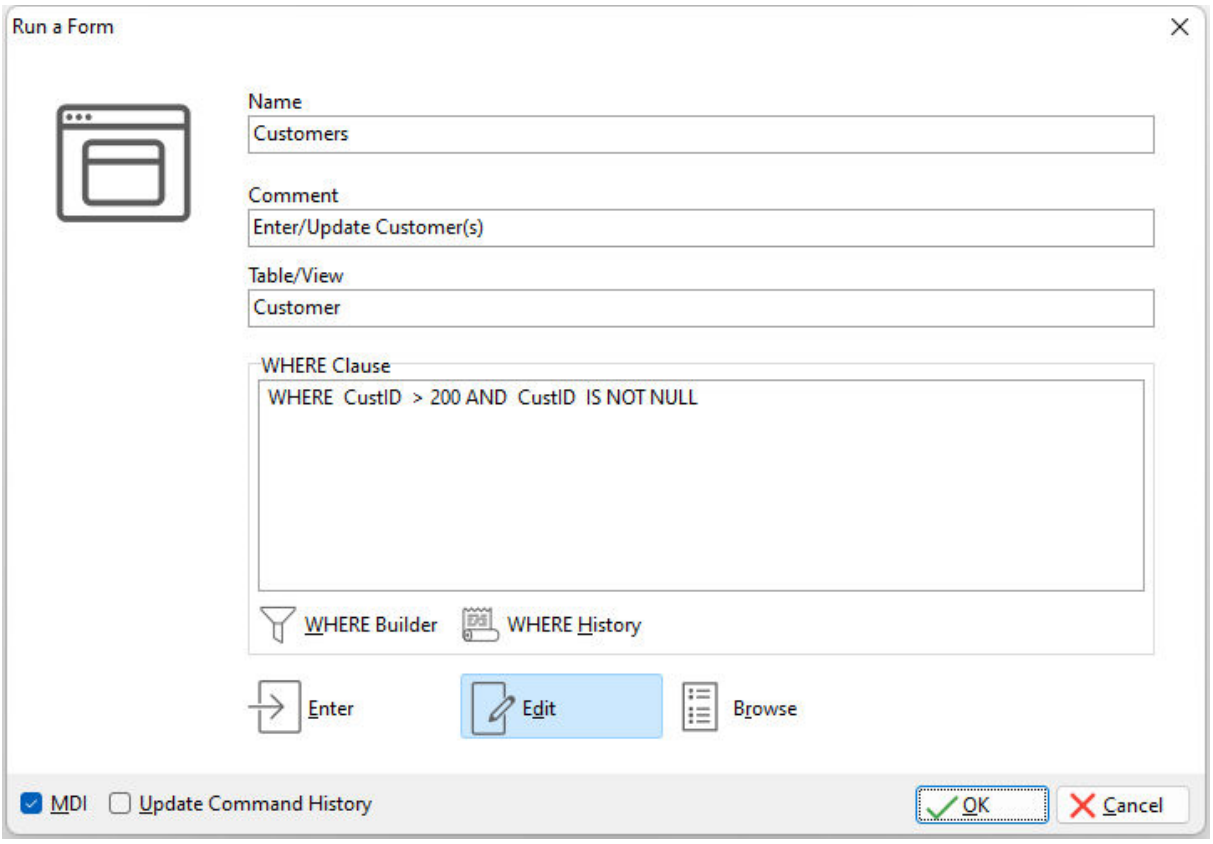
Using the R:BASE command syntax a modeless print preview screen can be launched with the PRINT and LBLPRINT commands. Each command supports an MDI parameter to specify if the preview is to be opened as a modeless window.

1.26.4 Running MDI Forms

A form can be launched in MDI mode from the user interface with the Database Explorer and main toolbar, or by using the R:BASE command syntax at the R> Prompt and within a command file/application.

When running a form within the user interface, an initial dialog will launch with several options including an "MDI" check box. If the MDI setting is set to ON, a check will already be placed in the box when the dialog opens.

Copyright © 1982-2024 R:BASE Technologies, Inc.



When MDI is checked, the form will be displayed as a modeless form. The form will appear in the center of the screen and look just as if the form was launched without MDI, but the difference is you will be able to continue accessing menus in R:BASE without having to close the form. If you click outside of the form and use other aspects of R:BASE, the form will be located behind the R:BASE program window. The focus can be returned to the MDI form by selecting "Window" > "MDI Windows" > "Forms" > "Form Name:" from the main Menu Bar. Or, the R:BASE program window can be normalized and moved aside to see the form.

Using the R:BASE command syntax, a modeless form can be launched with the BROWSE USING, EDIT USING, and ENTER commands. Each command supports an MDI parameter to specify if the form is to be opened as a modeless form. In addition to MDI, the AS ALIAS parameter is available in the syntax to specify an alias name for the instance of the MDI form. An alias name is available so that the form can be programmatically specified with the SETFOCUS command within the application.

1.26.5 Switching Between MDI Windows

When several MDI windows have been launched within an R:BASE instance, the focus can be switched between the MDI form by selecting "Window" > "MDI Windows" from the main Menu Bar.

The R:BASE program window can also be normalized and moved aside to see the windows.

1.26.6 Commands With MDI Support

R:BASE provides several commands that enable users to display and manipulate modeless windows.

- BROWSE
- BROWSE USING
- CLOSEWINDOW
- EDIT

Copyright © 1982-2024 R:BASE Technologies, Inc.

- EDIT USING
- ENTER
- GETPROPERTY
- LBLPRINT
- PRINT
- REFF
- SETFOCUS

1.26.7 Functions With MDI Support

R:BASE provides functions to assist in using modeless windows.

(IFWINDOW('*windowname*'))

Returns 1 if a form with the *windowname* is open, 0 if not. *Windowname* is the name given to the instance of an MDI form started with the "AS *alias*" option when using the ENTER, EDIT USING, or BROWSE USING commands.

(CVAL('MDI'))

Returns the value of the MDI operating condition (ON/OFF).

1.26.8 Sample Application

A sample application demonstrating the use of MDI forms, reports, and displaying data is available at the Sample Application Web page.

http://www.razzak.com/sampleapplications/

Document Name: **Designing MDI Applications**

1.27 Multi-User Guide

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

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

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

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

- Know your hardware. Have the best servers, workstations, switches, cabling and wireless nodes you can afford. Ensure that each is competently specified, configured, installed and maintained by someone who knows his or her craft, as well as the history of your system.
- Know your cabled Ethernet network. Same.
- Know your wireless Ethernet network. Same. But also, know when a Remote Desktop approach can overcome the performance limitations of wireless networks, especially within manufacturing environments where interference and signal degradation may degrade transmission of data.
- When designing your database schema, take full advantage of the relational capabilities of R:BASE for the linking and manipulation of data stored in multiple tables.

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Learn and apply the rules of normalization to your schema so that data is as "atomized" as possible, thus minimizing the resources required by any user at a given moment.
- Take full advantage of temporary tables and views in your coding to reduce the number and frequency of (otherwise avoidable) "hits" on permanent tables.

Topics

1.27.1 Multi-User Concurrency Settings

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

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

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

for all users								
Different value for users	NO	NO	NO	YES	YES	YES	YES	YES
Set once per user	YES	YES	YES	NO	NO	NO	NO	NO
May change per user	NO	NO	NO	YES	YES	YES	YES	YES

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

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

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

Related Topics:

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

1.27.1.1 MULTI

Operating Condition

Syntax: SET MULTI ON/OFF

Default: OFF

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

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

1.27.1.2 STATICDB

Syntax: SET STATICDB ON/OFF

Default: OFF

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

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

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

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

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

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

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

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

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

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

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

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

1.27.1.3 FASTLOCK

Syntax: SET FASTLOCK ON/OFF

Mode: Multi-user and STATICDB

Default: OFF

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

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

The following command lines set STATICDB and FASTLOCK correctly.

```
SET STATICDB ON
SET FASTLOCK ON
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
CONNECT concomp
```

1.27.1.4 PAGELOCK

Syntax: SET PAGELOCK ON/OFF

Mode: Multi-user

Default: ON

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

The settings for PAGELOCK are:

- **ON -** R:BASE uses page locking or row locking as appropriate. When PAGELOCK is ON and two or more users are updating rows within the same page of data, R:BASE only lets the first user update rows--the other users are locked out until the first user's update has been completed.
- **OFF -** R:BASE uses a fast row-locking method where only row locking is applied with no page locking. When PAGELOCK is OFF, you can lock rows of data instead of locking a page of data. You increase multi-user performance when PAGELOCK is OFF. And even more so when STATICDB and FASTLOCK are on.

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

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

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

Notes:

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

Example:

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

1.27.1.5 ROWLOCKS

Syntax: SET ROWLOCKS ON/OFF

Default: ON

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

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

1.27.1.6 VERIFY

Syntax: SET VERIFY COLUMN/ROW

Default: COLUMN

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

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

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

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

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

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

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

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

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

1.27.1.7 WAIT

Syntax: SET WAIT value

Range: 0 to 16383 seconds

Default: 4

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

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

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

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

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

```
SET WAIT 20
```

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

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

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

1.27.1.8 INTERVAL

Syntax: SET INTERVAL value

Default: 5

Range: 0 to 9 tenths of a second

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

1.27.1.9 Table Locks

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

Type of Lock	Command	Description
Concurrency Control	EDIT EDIT USING ENTER	Prevents one user from accidentally overwriting another user's changes. These commands can access the same table simultaneously.
Row Lock	DELETE ROWS ENTER *form* INSERT *values* LOAD FROM *filespec* UPDATE	When SET ROWLOCKS is on, a lock is only applied to a row. When off, a table lock is in effect.
Table Lock	BACKUP* DROP RULES FORMS GRANT INSERT *subselect* RBLABELS LOAD *from filespec* REPORTS RESTORE REVOKE RULES SET LOCK ON UNLOAD*	Must wait for commands that obtain table and database locks. Once obtained, this lock excludes all other concurrency control and locking until these commands have finished.
Full Database Lock	BACKUP ALL RELOAD UNLOAD ALL	All tables in the database are locked.
Database Schema Lock	ALTER TABLE CREATE SCHEMA CREATE TABLE CREATE VIEW CREATE INDEX DROP INDEX DROP COLUMN DROP TABLE DROP VIEW INTERSECT JOIN PROJECT RENAME SUBTRACT UNION	Engages a full database lock preventing schema modifications, then releases the schema lock remaining in table lock.
Cursor Lock	OPEN *cursorname*	Stops database schema commands but allows table locks; acts as a table lock.

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

Database Lock	Wait	Wait	Wait	Wait	Quit	Wait
Schema Lock	Wait	Wait	Wait	Wait	Quit	Wait

1.27.1.9.1 Using SET LOCK to Set Exclusive Table Locks

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

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

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

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

1.27.1.9.2 Displaying Multi-User Locks

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

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

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

1.27.1.10 Other Multi-User Considerations

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

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

R:BASE Commands	Operating System Files Used	Access Limitations
CODELOCK	From ASCII to converted binary file	One user at a time can convert a given ASCII file
FORMS	Database files	One user at a time can modify a form in a given database.
REPORTS	Database files	One user at a time can modify a report in a given database.
GATEWAY	Database files	Data files for transfer to database, database files. One user at a time can use data files.

Copyright © 1982-2024 R:BASE Technologies, Inc.

RBDEFINE	Database files	One user at a time can create or change database structure.
RBEDIT	ASCII file	One user at a time can create or edit a file.
LAUNCH	External programs	All users can share the program, within the limitations of the external program.
ZIP	External programs	All users can share the program, within the limitations of the external program.

1.27.2 Preparing for Network Use

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

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

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

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

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

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

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

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

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

R:BASE Installation Options
Version Control
Security Software Exceptions
Preparing the Application
Customizing the End User's Computer

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.27.2.1 R:BASE Installation Options

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

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

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

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

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

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

1.27.2.1.1 Client Installation

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

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

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

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

See Also Customizing the End User's Computer

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.27.2.1.2 Server Installation

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

Using this method, you will insert the R:BASE CD-ROM into the server's CD-ROM drive and run the installer. Or, if the installer was made available through a download, you would run the "Setup.exe" while physically sitting at the server. Or, you can run the "Setup.exe" on a workstation and change the "Destination Folder" to a shared network directory during the Setup process, but this option prevents the proper installation of the R:BASE ODBC driver. If you have a R:BASE application which uses the R:BASE ODBC driver, you must run the installer while at the server.

When running the installer, the "Components" screen will display a "Server Installation" option that must be selected. For any workstation where R:BASE development will be actively performed, the R:BASE program should be installed on the local computer. This will allow access to the help manuals and documentation.

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

Server Installation Considerations

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

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

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

- Launching the R:BASE Compiled Help (.CHM) - For those who are running the R:BASE "development" environment from a network installation and will launch the compiled help files (.CHM) from the network drive, the help files will not display properly. The usual message displayed is a "Action canceled" screen. This is the result of a Microsoft Security update that prevents users from running compiled Help files (.CHM) on network drives, as it may pose a threat. R:BASE Technologies, Inc. has no control over how the compiled help files are displayed, as this is an operating system update. Users may otherwise store and launch the R:BASE Help files on the local hard drives of the computers. This would be beneficial to any R:BASE user with the program installed on the network, and to any custom application where they to run their own compiled help.

See Also Customizing the End User's Computer

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.27.2.2 Version Control

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

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

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

1.27.2.3 Security Software Exceptions

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

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

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

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

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

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

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

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

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

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.27.2.4 Preparing the Application

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

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

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

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

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

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

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

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

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

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

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

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

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

1.27.2.4.1 Temporary Scratch Files

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

Syntax: SET SCRATCH <parameter>

Default: TMP

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

To set SCRATCH to TMP, use the following:

```
SET SCRATCH TMP
```

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

```
SCRATCH TMP
```

1.27.2.4.2 Customizing the Configuration File

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

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

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

1.27.2.4.3 Creating the Startup File

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

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

Startup files can be specified in two ways:

1. **Desktop Shortcut Properties**

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

2. **R:BASE Configuration File**

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

Notes:

- If a startup file is specified in both the configuration file and the desktop shortcut, R:BASE executes the commands in the startup files specified in the configuration file first, then any file specified after the R:BASE executable.
- If an RBASE.DAT file is located in the "start in" or "working" directory upon launching R:BASE, the program will always automatically run this file.
- Do not include the following commands in a startup file:
  - GATEWAY
  - ZIP ROLLOUT
- If you are using CodeLocked procedure files (APP, APX) as the source for your R:BASE custom application, you would create an R:BASE startup file to initialize the procedure file. The startup file would only require two commands, RUN and EXIT:

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

Examples:

Example 01:

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

Example 02:

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

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

```
-- MyApp.DAT Startup Application File
-- Start Fresh
   CLEAR ALL VARIABLES
LABEL StartFresh
   DISCONNECT
   SET QUOTES=NULL
   SET QUOTES='
   SET DELIMIT=NULL
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
    RETURN
-- End here ...
```

1.27.2.5 Customizing the End User's Computer

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

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

Client Installation

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

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

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

Example:

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

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

Example:

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

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

Server Installation

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

Example:

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.27.2.5.1 Startup Parameter for Server Environments

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

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

- Force "View As List" display, to omit displaying column details for objects
- UINOTIF is set to OFF, to turn off refresh notifications
- DB Explorer, does not enable Show Slave Table
- DB Explorer, does not enable indicator For Compressed Forms/Reports/Labels
- DB Explorer, does not enable Show Row Numbers
- DB Explorer, disables custom Explorer Appearance settings
- For Startup Options, does not display product splash screen

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

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

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

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

1.27.3 Increasing Application Performance

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

1.27.3.1 R:BASE Themes

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

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

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

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

Load Theme Example:

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

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

Release Theme Example:

```
PROPERTY RELEASE_THEME NewLuna TRUE
```

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

Change Theme Example:

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

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

1.27.3.2 Index Efficiency

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

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

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

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

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

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

3. Indexing column where NULL values exist

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

4. Lengthy text column indexes

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

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

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

6. Ascending/Descending "ORDER"

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

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

7. Not using indexes where needed

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

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

1.27.3.3 Optimizing Command Syntax Techniques

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

- **Group similar commands**

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

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

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

  The following command combines the four UPDATE commands into one.

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

- **Minimize disk access**

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

- **Use WHILE loops**

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

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

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

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

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

- **Use forward GOTO searches**

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

- **Predefine variables**

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

- **Reduce redundancy**

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

- **Eliminate rules checking**

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

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

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

1.27.3.4 Finding Minimum and Maximum Values

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

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

```
SET VAR t1 TIME = NULL
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

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

--Example 2 - Faster
SET VAR t1 = (.#TIME)
SET VAR vMin INTEGER
SET VAR vMax INTEGER
COMPUTE vMin AS MIN Altitude, vMax AS MAX Altitude +
  FROM Airports
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

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

1.27.3.5 Surrounding the WHERE Clause with Parentheses

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

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

1.27.3.6 Accumulating Data with SELECT

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

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

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

DROP TABLE Totals
CREATE TABLE Totals (State TEXT 15, Length INTEGER)
DECLARE c1 CURSOR FOR SELECT State, StCode +
FROM States +
WHERE State IN ('Washington', 'Oregon', 'California')
OPEN c1
WHILE SQLCODE = 0 THEN
FETCH c1 INTO vState INDICATOR ivState, +
  vStCode INDICATOR ivStCode
IF SQLCODE <> 0 THEN
BREAK
ENDIF
DECLARE c2 CURSOR FOR SELECT Runway +
FROM Airports WHERE StCode = .vStCode
OPEN c2
WHILE SQLCODE = 0 THEN
FETCH c2 INTO vRunway INDICATOR ivRunway
IF SQLCODE = 0 THEN
SET VAR vLength = (.vLength + .vRunway)
ENDIF
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
ENDWHILE
INSERT INTO Totals VALUES (.vState, .vLength)
DROP CURSOR c2
ENDWHILE
DROP CURSOR c1
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

--Example 7 - Faster
SET VAR t1 = (.#TIME)
DROP TABLE Totals
CREATE TABLE Totals (State TEXT 15, Length INTEGER)
DECLARE c1 CURSOR FOR SELECT State, StCode FROM States +
WHERE State IN ('Washington', 'Oregon', 'California')
OPEN c1
WHILE SQLCODE = 0 THEN
FETCH c1 INTO vState INDICATOR ivState, +
  vStCode INDICATOR ivStCode
IF SQLCODE <> 0 THEN
BREAK
ENDIF
COMPUTE vLength AS SUM Runway FROM Airports +
WHERE StCode = .vStCode
INSERT INTO Totals VALUES (.vState, .vLength)
ENDWHILE
DROP CURSOR c1
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

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

1.27.3.7 Using Nested Cursors

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

In Example 9, the DECLARE CURSOR is placed inside the WHILE loop of the first cursor. This requires the second DECLARE CURSOR command to execute every time the first command finds a row. In Example 10, both DECLARE CURSOR commands are executed at the start of the program, and the second

Copyright © 1982-2024 R:BASE Technologies, Inc.

DECLARE CURSOR command is opened inside the WHILE loop of the first command. By restricting the second DECLARE CURSOR from executing as often, this example runs much faster.

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

--Example 9 - Slow
SET VAR t1 = (.#TIME)
DECLARE c1 CURSOR FOR SELECT State, StCode +
FROM States +
WHERE State IN ('Washington', 'Oregon', 'California')
OPEN c1
WHILE SQLCODE = 0 THEN
FETCH c1 INTO vState INDICATOR ivState, +
  vStCode INDICATOR ivStCode
IF SQLCODE <> 0 THEN
BREAK
ENDIF
DECLARE c2 CURSOR FOR SELECT Runway +
FROM Airports WHERE StCode = .vStCode
OPEN c2
SET VAR vLength = 0
WHILE SQLCODE = 0 THEN
FETCH c2 INTO vRunway INDICATOR ivRunway
IF SQLCODE = 0 THEN
IF vRunway > 5000 THEN
SET VAR vRunway = (.vRunway + 100)
ELSE
SET VAR vRunway = (.vRunway + 50)
ENDIF
UPDATE Airports SET Runway = (.vRunway) +
WHERE CURRENT OF c2
ENDIF
ENDWHILE
DROP CURSOR c2
ENDWHILE
DROP CURSOR c1
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

1.27.3.8 Displaying Messages in Long Tasks

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

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

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

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
  SET VAR vcounter = (.vcounter + 1)
  PROCESSMESSAGE
ENDWHILE
CLEAR VARIABLE vcounter
CLS
```

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

```
PROPERTY SaveButton ENABLED FALSE
PROPERTY ProgressPanel VISIBLE TRUE
--Step #1
PROPERTY ProgressPanel CAPTION 'Working on stuff #1...'
PROCESSMESSAGE
...
...
...
--Step #2
PROPERTY ProgressPanel CAPTION 'Working on stuff #2...'
PROCESSMESSAGE
...
WHILE ...
  ...
      ...
      --if a single iteration is quick enough, PROCESSMESSAGE can be called after a
few iterations
      PROCESSMESSAGE
ENDWHILE
...
--Step #3
PROPERTY ProgressPanel CAPTION 'Working on stuff #3...'
PROCESSMESSAGE
...
...
...
PROPERTY ProgressPanel VISIBLE FALSE
PROPERTY SaveButton ENABLED TRUE
```

1.27.4 Increasing Performance in Forms

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.27.4.1 Moving Form Lookup Variables into Custom EEPs

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

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

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

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

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

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

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

```
-- On Exit EEP
SET VAR vClientFirstName = NULL
SET VAR vClientLastName = NULL
SET VAR vClientCompany = NULL
SET VAR vClientAddress1 = NULL
SET VAR vClientAddress2 = NULL
SET VAR vClientCity = NULL
SET VAR vClientState = NULL
SET VAR vClientZipCode = NULL
SET VAR vClientPhone = NULL
SET VAR vClientFax = NULL
SET VAR vClientEmail = NULL
SELECT +
   CFirstName, +
   CLastName, +
   CCompany, +
   CAddress1, +
   CAddress2, +
   CCity, +
   CState, +
   CZipCode, +
   CPhone, +
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
   CFax, + +
   CEmail +
INTO +
   vClientFirstName INDIC iv1, +
   vClientLastName INDIC iv1, +
   vClientCompany INDIC iv1, +
   vClientAddress1 INDIC iv1, +
   vClientAddress2 INDIC iv1, +
   vClientCity INDIC iv1, +
   vClientState INDIC iv1, +
   vClientZipCode INDIC iv1, +
   vClientPhone INDIC iv1, +
   vClientFax INDIC iv1, +
   vClientEmail INDIC iv1 +
FROM Client WHERE ClientID = .vClientID
RECALC VARIABLES
RETURN
```

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

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

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

```
SET VAR vTPPContactName = NULL
SET VAR vTPPCompany = NULL
SET VAR vTPPConract = NULL
SET VAR vTPPCertificate = NULL
SET VAR vTPPNotes = NULL
SELECT +
   TPPContactName, +
   TPPCompany, +
   TPPContract, +
   TPPCertificate, +
   TPPNotes ,+
INTO +
   vTPPContactName INDIC iv1, +
   vTPPCompany INDIC iv1, +
   vTPPConract INDIC iv1, +
   vTPPCertificate INDIC iv1, +
   vTPPNotes INDIC iv1 +
FROM ThirdParty WHERE TPP_ID = .vTPP_ID
RECALC VARIABLES
RETURN
```

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
21 : CURRENCY   vInvoiceTotal = (.vSubTotal + .vFreight + .vSalesTax)
```

Note:

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

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

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

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

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

1.27.4.2 Command Syntax in EEPs

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

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

1. In the Form Designer, select one of the controls which references a command file EEP.
2. Right click on the control and select "Properties'.
3. From the "EEPs" tab, select the "Edit EEP..." button, which will display your EEP code in the R:BASE Editor.
4. Use the keyboard shortcut [Ctrl] + [A] to highlight the entire contents of the command file EEP.
5. Use the keyboard shortcut [Ctrl] + [C] to copy the contents to the Windows clipboard.
6. Close the R:BASE Editor window to return to the control properties dialog.
7. Select the "Edit Custom EEP..." button, which will open the R:BASE Editor to store the code, only this time the commands will be stored within the form itself.
8. Use the keyboard shortcut [Ctrl] + [V] to paste the entire contents of the command file EEP.

Copyright © 1982-2024 R:BASE Technologies, Inc.

9. Select the "OK" button to return to the control properties dialog.
10. Remove the EEP file name from the "Custom" field so the EEP does not fire from the command and the Custom EEP, and only the Custom EEP.

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

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

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

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

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

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

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

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

```
PAUSE 3 USING 'Calculating ... Please Stand By ...' +
CAPTION 'Calculating ...' ICON APP +
OPTION GAUGE_VISIBLE ON +
|GAUGE_COLOR [R218,G228,B246] +
|GAUGE_INTERVAL 10 +
|MESSAGE_FONT_NAME VERDANA +
|MESSAGE_FONT_SIZE 10 +
|MESSAGE_FONT_COLOR BLUE +
|THEMENAME Razzmatazz
 -- Put your code, that takes a lot of time, here ...
 -- Use CLS command to clear the PAUSE 3 dialog
CLS
RETURN
```

1.27.4.3 Disable RBTI Variable Processing

An added PROPERTY APPLICATION is available parameter to specify if "RBTI Variable" (RBTI_*) processing is enabled in forms. If perhaps the variables are seldom used in an application, the processing may be turned off, as the variables are assigned in nearly all forms aspects. The property is

Copyright © 1982-2024 R:BASE Technologies, Inc.

ON by default. To take advantage of the optimization, set the value to OFF upon the application startup. Then, turn it ON only in forms that use the RBTI_* variable values. Make sure the property is turned OFF again when the form is closed.

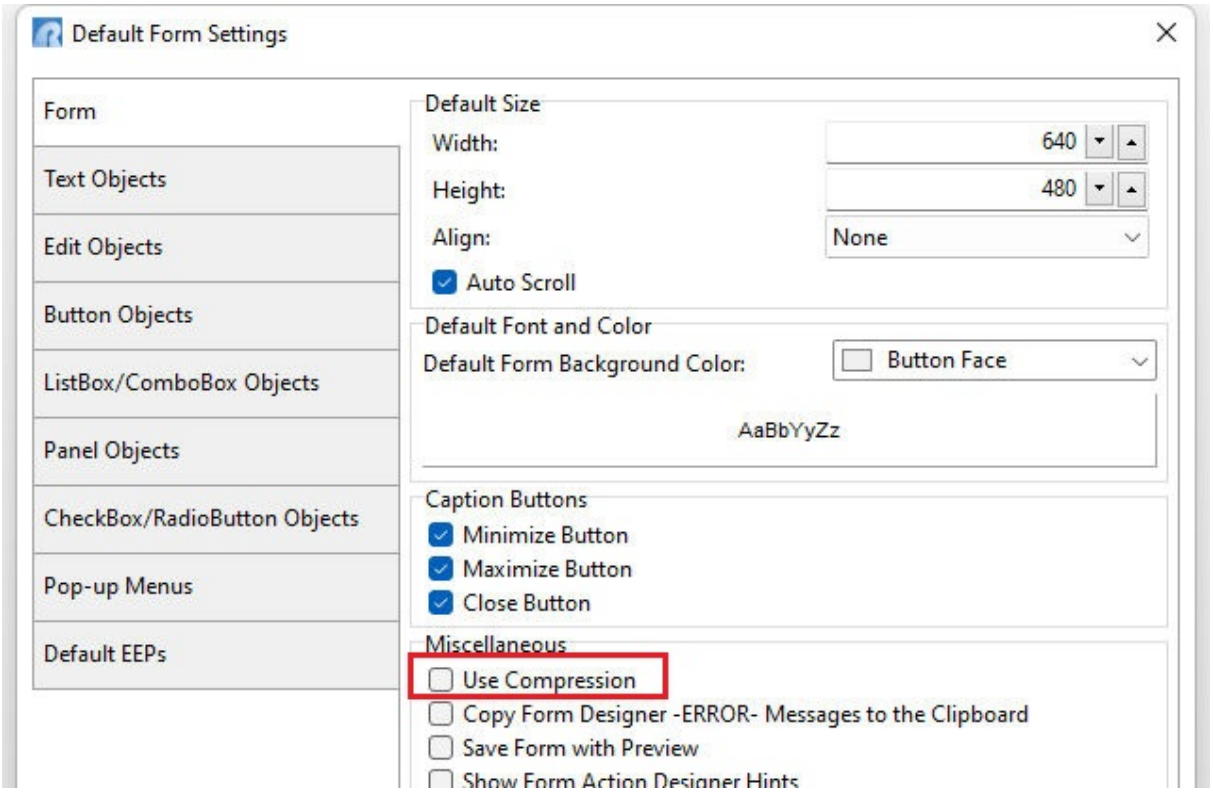
```
PROPERTY APPLICATION CAPTURE_RBTI_FORM_VARS OFF
```

1.27.4.4 R:BASE Form Compression

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

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

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



1.27.4.5 DB Grid and Enhanced DB Grid

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

A. DB Grid "Power"

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

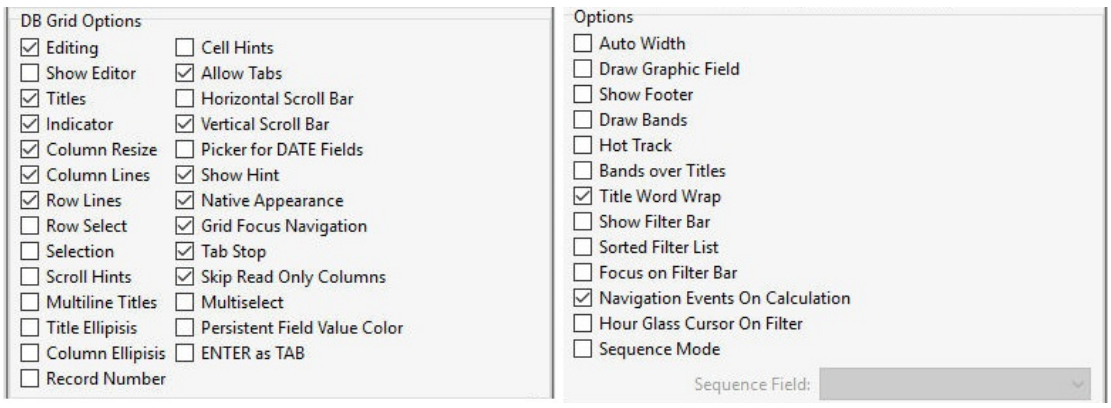
  You can also take advantage of the following RBTI Form Variables:

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

- There are **fourteen** different EEPs available within Enhanced DB Grids.

  - On Entry into DB Grid
  - On Exit from DB Grid
  - On Entry into DB Grid Column
  - On Exit from DB Grid Column
  - On DB Grid Column Moved
  - On DB Grid Cell Click
  - On DB Grid After Cell Click
  - On Double Click
  - On Group Expression
  - On Column Widths Change
  - On Filter Change
  - On Mouse Enter
  - On Mouse Leave
  - On Top Left Change

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



Copyright © 1982-2024 R:BASE Technologies, Inc.

B. DB Grid "User Friendliness"

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

C. Grid "Visual Illustration"

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

1.27.4.6 Fewer Results in Lookup Controls

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

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

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

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

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

1.27.4.7 Fewer Results in Pop-up Menus

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

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

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

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

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

   Examples of dynamic WHERE Clauses:

   Example 01:

```
-- Dynamic WHERE Clause (Company LIKE)
Message:
Enter First Few Characters of Company
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
WHERE Clause:
WHERE Company LIKE '&%' ORDER BY Company
```

Example 02:

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

Example 03:

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

1.27.5 Operating in Single-User Mode

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

To operate in single-user mode:

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

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

1.28 Problem Solving in R:BASE

Identifying the problem is the first, but not necessarily the easiest, step in solving problems. Often a software problem starts as "*The program doesn't work*," or "*it worked fine the last time I did it.*" Obviously, this sentiment isn't identifying the real problem. The challenge in finding a solution, is determining the real problem. Once the problem is identified, the solution follows.

Today's personal computer is an environment that can spawn untold problems-a complex piece of hardware with processors, disk drives, keyboards, monitors and other devices connected by cables and more cables. Accompanying the hardware is a complex operating system controlling the different devices, and above all is the application software, for example, R:BASE with its relational tables, forms, reports and programs. Many times a "computer problem" is not a problem at all, but simply a person's inability to remember all the processes and possibilities. Computers are the most obedient creatures around. They do exactly what they're told, nothing more and nothing less.

To get at the core of a problem, you need to eliminate some of the possible causes; fortunately, the process of elimination moves quickly. The comparison technique of problem solving works well and quickly with computer related problems to eliminate causes until only the actual problem remains.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.28.1 Making a Comparison

Compare the situation that has the problem to a similar situation or environment without the problem, and you should be able to find a difference. The difference leads to the cause of the problem, or at least to an area to be investigated. You must, however, make the comparison before you can see the differences that form the clues to the problem and thus the solution.

Compare the environment; the difference may be here. For example:

"It works on Joe's PC, but not on mine."
"I just upgraded the operating system."
"I bought a new hard drive."
"I upgraded to a newer version of R:BASE."
"I swapped memory."
"I just installed a new operating system update."

Compare the processes; the difference might be in the action or activity. For example:

"My program is running slower than it used to."
"The totals in my report are much larger than they used to be."
"My program used to return to the menu, now it leaves me at an R> Prompt."
"It only happens the second time I do it."
"I modified my program, just a little bit."

Make a time comparison; the difference may be connected to the time of day. For example:

"The network always slows down at lunch time."
"At 9:00 every morning I get kicked out of R:BASE."
"It only happens when Mary is printing the Sales Report."

Compare the problem with a similar situation. For example:

"It prints to the screen, but not to the printer."
"It works from the R> prompt, but not in my application."
"It works in the sample database, but not in my database."
"It works when my database is local, not when it is on the network."

Sometimes it is hard to identify a difference. In fact there can be many situations with no apparent difference or change at all. This may be the time to contact someone who can offer a solution or suggest other areas to check.

1.28.2 Comparing R:BASE Installations

Below is a list list of R:BASE areas to compare between computers when trying to isolate a problem.

Version and Build number

Database
- Name
- Size of files
- Location (local or network)
- Passwords
- AUTOCHK

Tables
- Name
- Number of Rows
- Columns
- Name
- Autonumbered
- Indexed (single column, multi-column)
- Computed
- Constraints
- Rules

Copyright © 1982-2024 R:BASE Technologies, Inc.

Views
- Select command
- Linking columns
- Where clause
- Correlation names
- Outer Join
- Union
- Place holders
- Aliases

Settings
- Messages
- Error Messages
- TRACE
- WHILEOPT
- SCRATCH
- ZERO

Multi-User Settings
- FASTLOCK
- INTERVAL
- LOCK
- MULTI
- NAME
- QUALCOLS
- REFRESH
- ROWLOCKS
- STATICDB
- TIMEOUT
- VERIFY
- WAIT
- RBADMIN

Command Files
- Command syntax
- Comments
- Application environment (forms, code, EFF, RBA)
- CodeLock
- Start up files
- Variables
  - Name
  - Data type
  - Values
  - Expressions
  - Functions

Forms
- Add data or edit data
  - Where clause
  - Order by clause
  - Form settings
  - Number of Tables/Views
- Table settings
  - Column relationships
  - Region
  - Tiered
- Field locations
- Field settings
  - Default values
  - Pop ups
- Entry Exit Procedures (see Command Files)
  - Field level
  - Row/Table level
  - Form-in-a-form

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Custom Form Actions
- Lookups (same-table/other-table)
- Variables
  - Column/variable definition
  - Dependent
  - Values
  - Expressions
  - Functions

Reports
- Driving Table/View
  - Column relationships
- Print command
- Where clause
- Order by clause
- Variables
  - Lookups
  - Expressions
  - Functions
  - Data type
  - Order
  - Value
  - Calculate On
- Report Actions
- Custom EEPs
- Breakpoints
  - Columns
  - Reset variables
  - Form feeds
- Page Setup
  - Paper Size
  - Paper Source
  - Layout
  - Margins
- Report Bands
  - Report/Page/Break Headers
  - Detail lines
  - Break/Page/Report Footers
- Lines/boxes
- Column/Variable locations
- Pass Setting
- Cache Pages

1.28.3 Comparing Hardware and Environment

Computer
- Brand of the computer
- Type of CPU
- Amount of memory
- Stand-alone or on a network
- Size of the hard drive(s)
- Free (available) disk space
- Network Interface Card
- Video driver
- Screen saver

Operating System
- Version
- Updates
- ODBC

Printer
- Type of printer
- Brand of printer

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Parallel, USB, network printer connection
- Print spooler
- Printer driver

Network System

- Permissions
- Brand
- Version
- Search mappings or path
- Group/permission settings
- File attributes
- Dedicated server

1.28.4 Before You Call

Before you call someone, take a look at any differences you have identified. The difference might Indicate who to confer with on possible causes. If the problem occurred when you changed hardware, then call your hardware technician. If the problem occurred when you switched operating systems, then contact the manufacturer of the operating system. You should contact R:BASE Technologies Technical Support at mailto:support@rbase.com for issues related to your R:BASE program. An experienced support technician has a good idea where to look for solutions based on the description of the problem. Our technicians can help you to identify differences and make comparisons for arriving at a solution.

The list of items to compare or check is long and by no means complete. You do not have to compare every item when problem solving. Compare the areas involving your problem to those areas where you may recently have made some type of change. Often finding one difference or change will lead to the next comparison to make and help you solve the problem.

When you have identified those things that have changed, you should be very close to the cause of the problem. Now it is time to check the R:BASE help documentation for the information that might offer a solution, or resources available to you.

In-line Help [F1] included with the product:

- RBG.chm
- Forms.chm
- Reports.chm
- Applications.chm
- EForms.chm
- QBE.chm
- Labels.chm
- Tutorial.chm
- Gateway.chm
- DataBrowseEdit.chm
- DBExplorer.chm
- Codelock.chm
- RBDefine.chm
- RBEdit.chm
- RPrompt.chm
- Trace.chm
- STProcedure.chm

PDF documentation included with the product:

- Command Index
- Function Index
- Reference Index
- Database Maintenance
- Database Conversion Guide
- Getting Started Guide
- Forms Manual
- How To Manual
- Programming In R:BASE
- Quick Installation Guide

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Beginners Tutorial
- What's New In R:BASE

When these resources have been exhausted, and you have not been able to find a solution, it is time to contact technical support. Because you have taken the time to research the problem, a support technician has an easier time discussing the solution, or offering alternatives and suggestions for your problem. When you make the proper comparison, you can find the difference that leads to a solution.

1.29 Purpose of a Rule

A data entry rule ensures that the data entered into a column meets the criteria you specify. You can use rules to do the following:

- Prevent duplicate information from being entered. For example, a new stock number cannot be the same as an existing one.
- Verify that the data being entered corresponds with data elsewhere in the database. For example, the product stock number must exist in a table before you enter a sales transaction for the product.
- Prevent a row from being deleted if it corresponds with data elsewhere in the database. For example, if a transaction refers to a customer, the customer cannot be deleted from the *customer* table.
- Define a value range. For example, when you enter a salary, it must be between $15,000 and $50,000. You can either specify a maximum value, a minimum value, or a range of values.

When you enter or edit data, R:BASE checks the data you enter with the corresponding rules. If the conditions of a particular rule are not met, R:BASE displays the error message defined for that rule and does not add the row. You can turn the rules setting on or off; the RULES setting must be on for R:BASE to check them.

As you use and modify the tables in your database, you might need to revise the data entry rules. For example, if you rename a column or table used in the WHERE clause of a rule definition, you must update the rule to reflect the change.

You can delete rules that no longer apply to your database. For example, delete rules that are associated with a column you have deleted from a table.

See also:

DROP Command
LIST Command
RENAME Command
RULES Command
SET RULES Command

1.30 R:BASE Debug Setting

The R:BASE Engine RDEBUG setting is available to create a log file to help understand possible issues when running R:BASE commands and using ODBC with foreign data sources. The log file will contain Engine Functions as R:BASE establishes and frees connections, executes SQL statements, retrieves data and values, controls transactions, and handles data and values.

To enable debugging, add the RDEBUG setting to your R:BASE product configuration file, which creates a log file of the R:BASE Engine Functions.

The following provides the supported use of the RDEBUG setting in the configuration file:

01. - Debugging is off

```
RDEBUG OFF
```

02. - Debugging is on, where a log file is created in C:\

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
RDEBUG ON
```

03. - Debugging is on with a file path and name. For the below, the rbengine.log file is created in the C:\Temp\RDEBUG\ folder.

```
RDEBUG ON C:\Temp\RDEBUG\rbengine.log
```

After the DEBUG setting has been turned ON or OFF, R:BASE must be restarted in order for the setting to be recognized.

Important: The debug setting and logging adds overhead to the R:BASE engine and performance will decrease. After logging has been captured for a desired event where an issue occurs, the debug setting should be set to OFF in the configuration file.

When RDEBUG is OFF the log file may remain in the configuration file.

```
RDEBUG OFF C:\Temp\RDEBUG\rbengine.log
```

The default location for the R:BASE configuration files is in "C:\Users\Public\RBTI", with RBENGINE11.CFG used for R:BASE 11.

1.31 Referential Data Integrity in R:BASE

By definition, referential integrity is a property of data which, when satisfied, requires any field in a table that is declared a foreign key can contain only values from a parent table's primary key.

In legacy R:BASE versions, data integrity and referential integrity had to be enforced through rules and programming. They are now a part of the database structure called **constraints** and are automatically enforced when defined. The constraints that can be defined include:

- Primary Key
- Foreign Key
- Unique Key
- Unique Index
- Not NULL

1.31.1 Constraints

Constraints provide automatic, database-wide data integrity and referential integrity. The not null constraint restricts data entry. No matter what method is used to enter data, the constraint verifies that the specified column has a value and the value is unique.

The primary key, foreign key, and unique key constraints provide both data integrity **and** referential integrity. The primary key column is automatically not null and unique. Deletions to a table with a defined primary key are automatically restricted if a referenced foreign key is defined. A value cannot be entered into a table with a foreign key unless that value exists in the referenced primary key table.

Constraints are similar to rules in that they indicate valid values or conditions that must be met in a table or column before data can be added, changed, or deleted. For instance, deleting a record that contains a value referred to by a foreign key in another table would break referential integrity.

Some constraints automatically define indexes when they are created; where they use the index to quickly check the condition. For example, a primary key is a constraint and an index is created on the primary key column(s). A constraint may use an index, but constraints and indexes are two different things. Constraints and indexes are created and listed separately. Constraints are commonly used to refer to primary and foreign keys.

Constraint Benefits Over Rules
Using rules to enforce these same data constraints required several different rules. You would need a rule to prevent duplicates (replaced by the primary key or unique constraint), verify a value rules (foreign key), and require a value rules (not null). In addition, you would need to define delete rules to

Copyright © 1982-2024 R:BASE Technologies, Inc.

prevent rows being deleted from the primary key table if there were matching rows in the foreign key table. If you wanted to prevent users from changing a primary key value that exists in the foreign key table, you would need to define an additional verify a value rule. Other constraint benefits over rules include:

- Easier to define
- Cannot be turned off, and are always in place
- Faster performance

1.31.1.1 Primary Key

A primary key is a column, or set of columns, that uniquely identify a row, meaning that each value in a primary key column is unique. As a constraint, the primary key prevents duplicate (non-unique) and null values from being entered into the column.

A primary key could be something like an employee id column in an employee table or it could be a combination of the customer's id and the customer's phone number.

Notes:

- Only one Primary Key can be defined per table.
- Defining a Primary Key automatically enforces "not NULL" and "unique" constraints on the column(s).
- A Primary Key cannot be defined if any one of the columns included in the desired key already have a NULL or duplicate values.
- R:BASE automatically builds an index on the specified column(s) when a Primary Key is defined.
- Every table should have a column or set of columns that identify a row, and (in a well-designed database) should have a Primary Key.
- A Primary Key definition is used instead of a Rule to prevent duplicates, and has the advantage of faster performance. It is recommended to delete the rules and indexes that are no longer needed before defining a Primary Key constraint.

1.31.1.2 Foreign Key

Like a primary key, a foreign key is a column or a group of columns. Foreign keys are also used to ensure that only valid data is entered into a column. A foreign key matches the values in a particular primary key or unique key constraint which is defined in a different table.

Primary and foreign keys must also match in terms of the specified columns. If you have a multi-column primary key, you can not have single column foreign keys reference it. If you have a multi column foreign key, it cannot reference a single column primary key. When a primary key is defined as more than one column, those columns are treated as a whole. Primary and foreign keys must match exactly. Thus, if your primary key is defined as text, then the corresponding foreign keys referencing it must also be text.

Primary and foreign keys automatically preserve referential integrity. You cannot delete a row from a table with a defined primary key if there are referenced foreign keys, thus you can never have detail records without a matching master record.

You can delete a row from a table with a foreign key. You cannot add a row to table with a foreign key defined unless the value entered matches a value in the referenced key table.

By default, users cannot update a primary key value if there are references, thus ensured that linking columns always match. The same applies for updating primary key values if there is not matching values in the referenced foreign key. However, implementing the r:base Cascade feature on primary keys will allow such changes. When a primary key is defined as a "cascading" primary key, the referenced foreign key values are automatically updated, deleted, or both. This allows the ability to update or delete referenced primary key values and retain the referential data integrity.

Notes:

- A Primary Key can exist without a Foreign Key, but a Foreign Key cannot exist without a Primary Key.
- A Foreign Key is always defined to reference a Primary Key or Unique Key.
- A Foreign Key automatically checks that the values in the Foreign Key exist in the referenced key.

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Many Foreign Keys can be defined in one table, and many Foreign Keys in different tables can reference the same Primary Key.
- Once Primary and Foreign Keys are defined accordingly, R:BASE automatically preserves the referential data integrity.
- A Foreign Key replaces a "Verify a value RULE".
- An index is automatically built when a Foreign Key constraint is defined.
- It is recommended to delete rules and indexes that are no longer needed before defining a Foreign Key constraint.

1.31.1.3 Unique Key

A unique key is a column or set of columns that uniquely identify a row; in other words, each value in a unique key column is unique. A unique key constraint prevents duplicate (non-unique) and null values from being entered into a table. The only difference between a unique key and a primary key is that you can define multiple unique keys per table.

Notes:

- A Unique Key constraint can replace a "Require a unique value" rule.
- A Unique Key constraint automatically builds an index.

1.31.1.4 Unique Index

A unique index is an index that uniquely identify a row. A unique index constraint prevents duplicate values from being entered into a table, and can prevent null values, if defined. The differences between a unique key and a unique index is that the unique key must be defined a Not NULL.

Notes:

- A Unique Index constraint can replace a "Require a unique value" rule.
- A Unique Index constraint can replace a "Require a value" rule.
- A Unique Index constraint automatically builds an index.

1.31.1.5 Not NULL

Placing a not null constraint on a column requires that the data in the column must contain a value, and cannot be null. This prevents users from adding a "blank" value. A not null constraint cannot be added if the column already contains null values.

Notes:

- A not null constraint can replace a "Require a value" rule. R:BASE does not build an index for a not null constraint, but since it stores the not null as part of the column definition, it is able to check the constraint faster than it could check the rule.
- If null values already exist, the values must first be edited to actual data values, before the not null constraint can be added.

1.31.1.6 Comparison of Constraints

Constraint	Unique	Not NULL	Indexed	Can Be Referenced	Multi-Column Constraint	Can Replace a Rule
Primary Key	Yes	Yes	Yes	Yes	Yes	Yes
Foreign Key	No	Intended	Yes	No	Yes	Yes
Unique Key	Yes	Yes	Yes	Yes	Yes	Yes
Unique Index	Yes	Optional	Yes	No	Yes	Yes
Not NULL	No	Yes	No	No	No	Yes

1.31.2 Cascade

Cascade is a referential integrity setting that can be applied to primary key tables, which maintains primary/foreign key relationships automatically. The Cascade options include:

- Update
- Delete

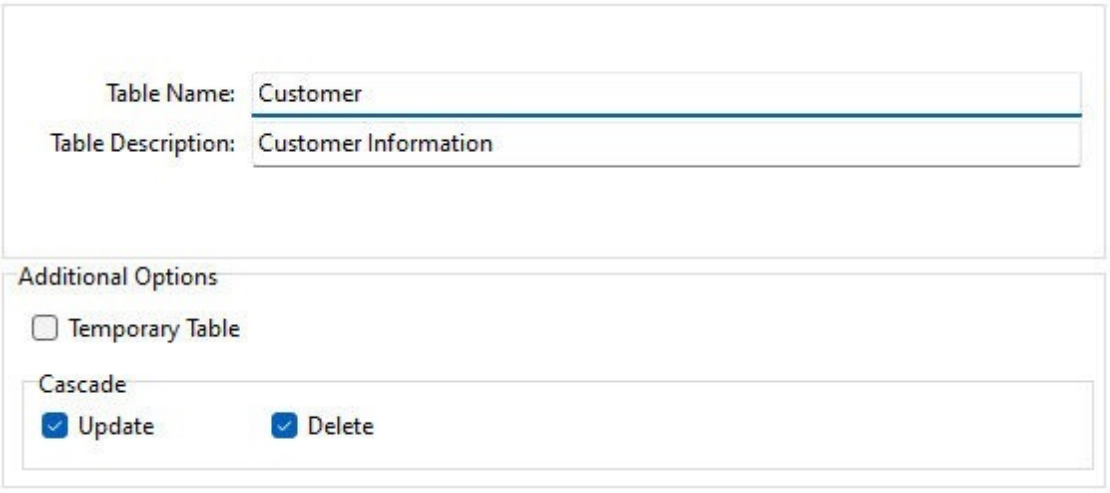
Copyright © 1982-2024 R:BASE Technologies, Inc.

Cascading updates enforce that when a primary key value is updated, the corresponding value in the foreign key table(s) will also be updated.
Cascading deletes enforce that when a primary key value is deleted, the corresponding value in the foreign key table(s) will also be deleted.

By not specifying either Update or Delete, both Cascade restrictions will be enforced upon the primary/foreign key tables and will prevent the values from being altered or deleted in the primary key table.

Separate Update and Delete data restrictions can allow a Cascade to be enforced for records that are updated, but not enforced when records are deleted, in order to avoid an accidental or undesired record delete. For example, if you either Update or Delete a primary key value from a table, the corresponding foreign key values are updated or deleted automatically. A Cascade can applied to Update, Delete or both to specific primary keys.

Cascade can be defined through the R:BASE command syntax using either the CREATE TABLE or ALTER TABLE commands. Cascade can also be set within the Data Designer when viewing the table properties.



Notes:

- Cascade can only be added to tables with defined Primary Keys.
- Cascade can be set to Update, Delete, or both.

1.31.3 Defining Constraints

Constraints are defined using the Data Designer or through the CREATE TABLE or ALTER TABLE commands.

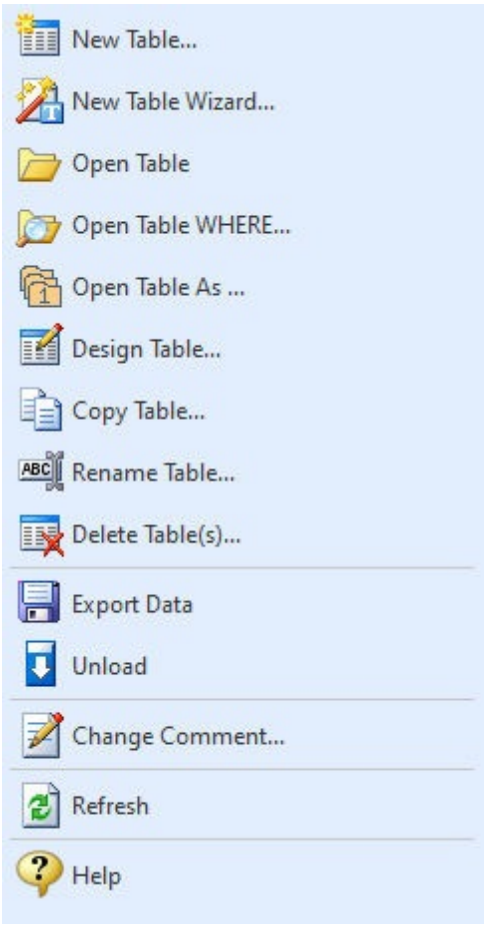
Before defining primary and foreign key constraints, decide which column or columns and which tables are best suited for the primary key. The column or columns selected should be the ones that uniquely identify the row from other rows in the table, and the table should be the one where that value is first entered into the database and is unique. Linking columns are generally good candidates for primary, foreign key constraints. For example, in the Concomp sample database, the empid column is used to link data in the Employee table with data in the Transmaster table and data in the Salesbonus table. Since a record is first entered into the Employee table, and when data is entered into the Transmaster or Salesbonus tables a matching record must already exist in the Employee table, the empid column in Employee becomes the primary key. The empid columns in Transmaster and Salesbonus become foreign keys referencing the Employee table primary key.

The direct benefit gained from designating empid as a primary key in the Employee table and foreign keys in the Transmaster and Salesbonus tables is the protection for your data from inadvertent changes. The empid value in Employee cannot be changed when there is a matching row in the Transmaster or

Copyright © 1982-2024 R:BASE Technologies, Inc.

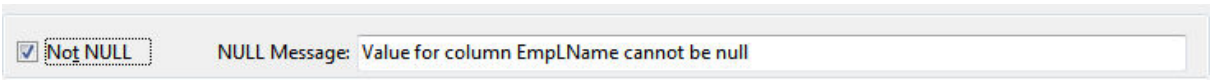
Salesbonus table. Rows cannot be deleted from either table. When adding data to Transmaster or Salesbonus you automatically require a matching value in Employee without having to define a rule.

The Data Designer
Launching the Data Designer to defined constraints can be performed directly within the Database Explorer window. Select the "Tables" option from the Group Bar to view the available options. Then, highlight a table, and select "Design Table...".



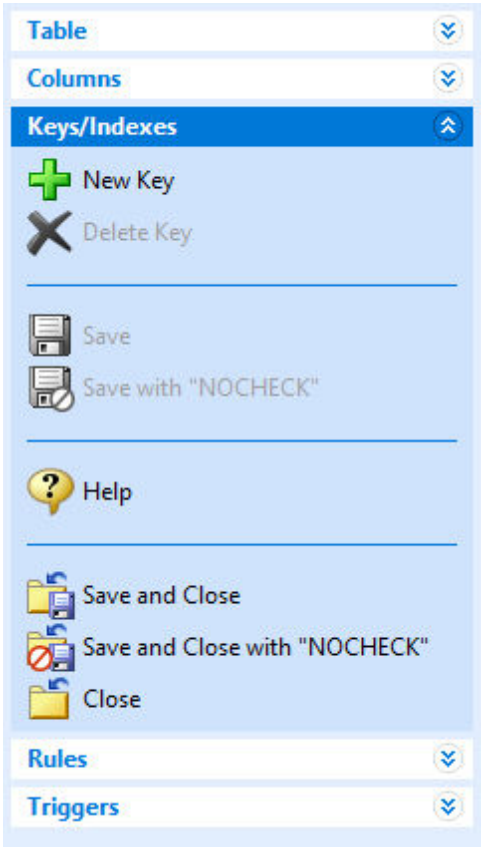
(Note: With an existing table selected, all of the above selections become enabled.)

The "**Columns**" option will display the defined columns for the table, where the "Not NULL" constraint can be enabled.



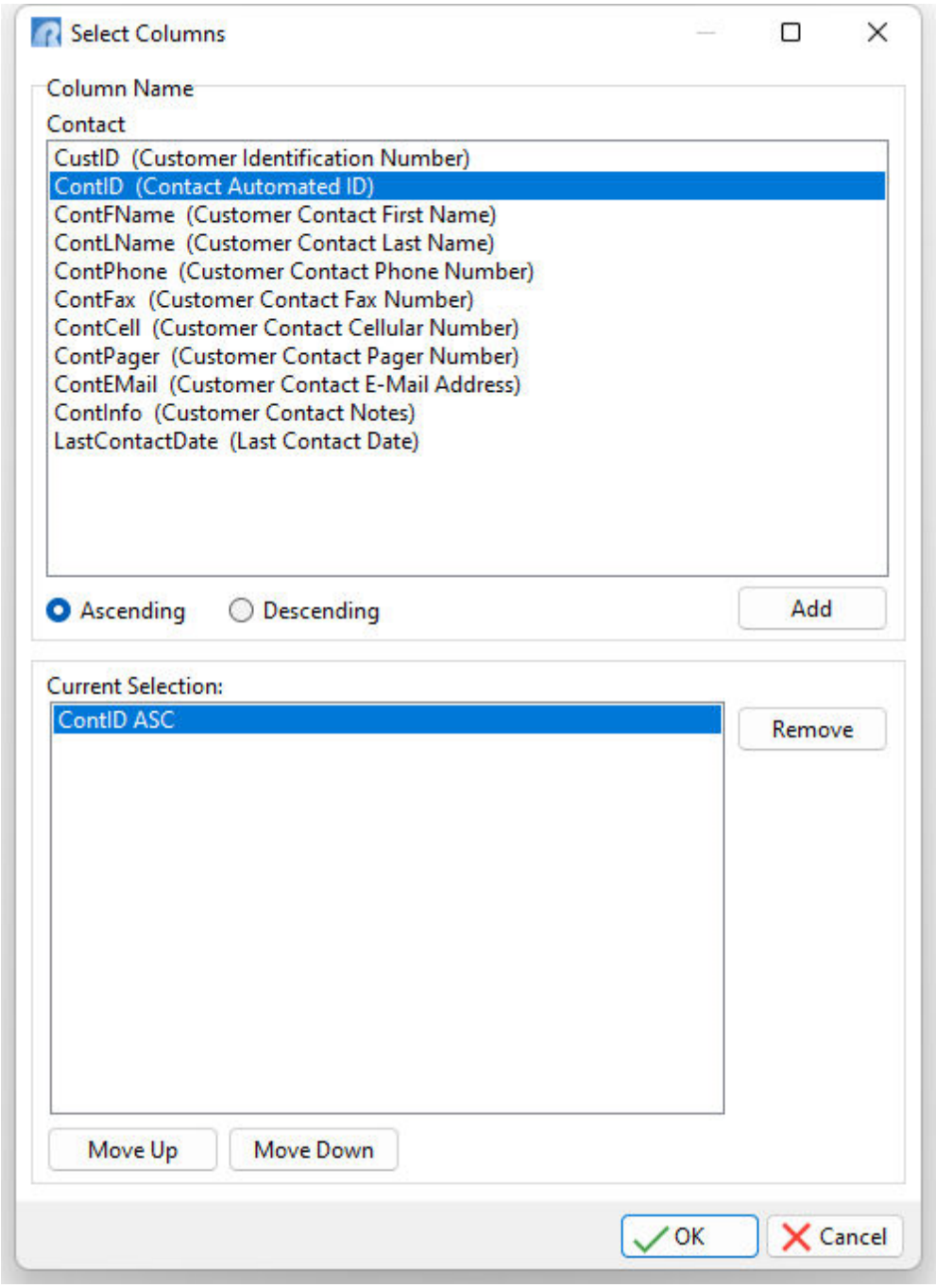
The "**Key/Indexes**" option allows users to create primary key, foreign key, unique key, and unique index table constraints. The "New Key" option allows users to create a new key/index

Copyright © 1982-2024 R:BASE Technologies, Inc.



When creating a new key, the "Select Columns" dialog appears first allowing you to select the column(s) and sorting order (Ascending/Descending) before defining the key.

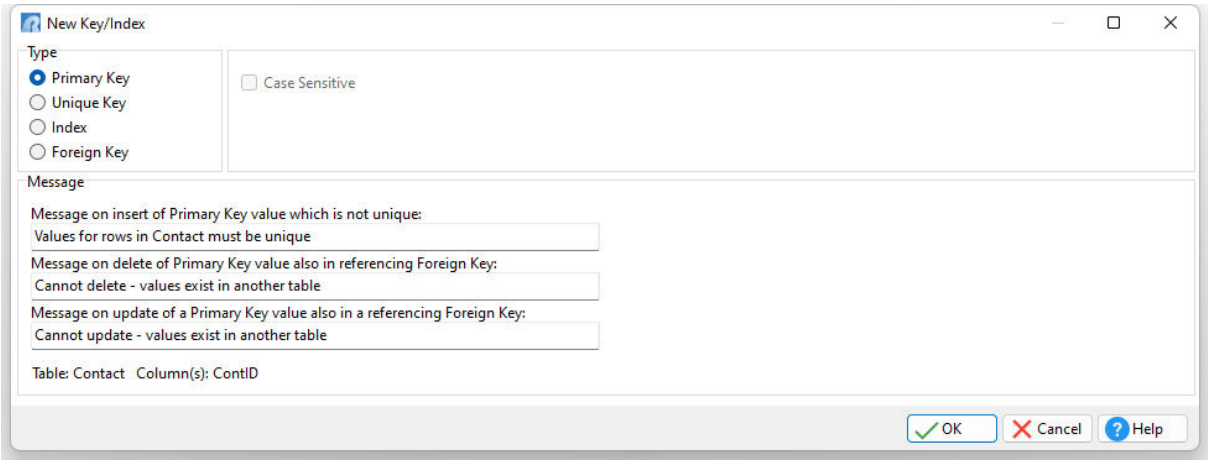
Copyright © 1982-2024 R:BASE Technologies, Inc.



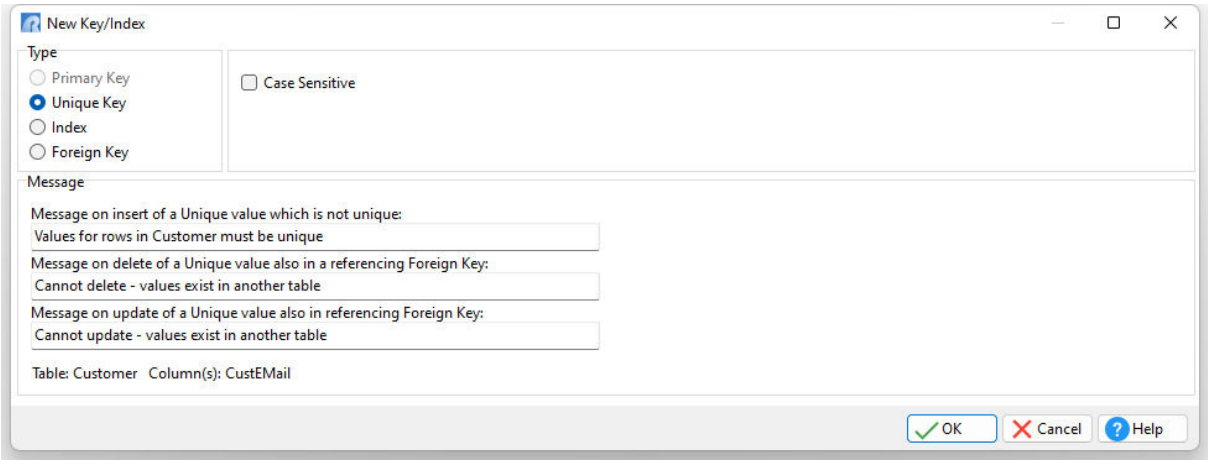
The "New Key/Index" dialog will display allowing for the creation of a primary key, foreign key, unique key, or index. If the Data Designer is launched for a table where a primary key is already defined, the option to specify a "Primary Key" type will be disabled. When adding a Primary Key, the options for "Not Null" and "Unique" are disabled by default and cannot be altered.

Custom constraint violation error messages can be specified for when an attempt is made to compromise the table's primary key referential integrity.

Copyright © 1982-2024 R:BASE Technologies, Inc.

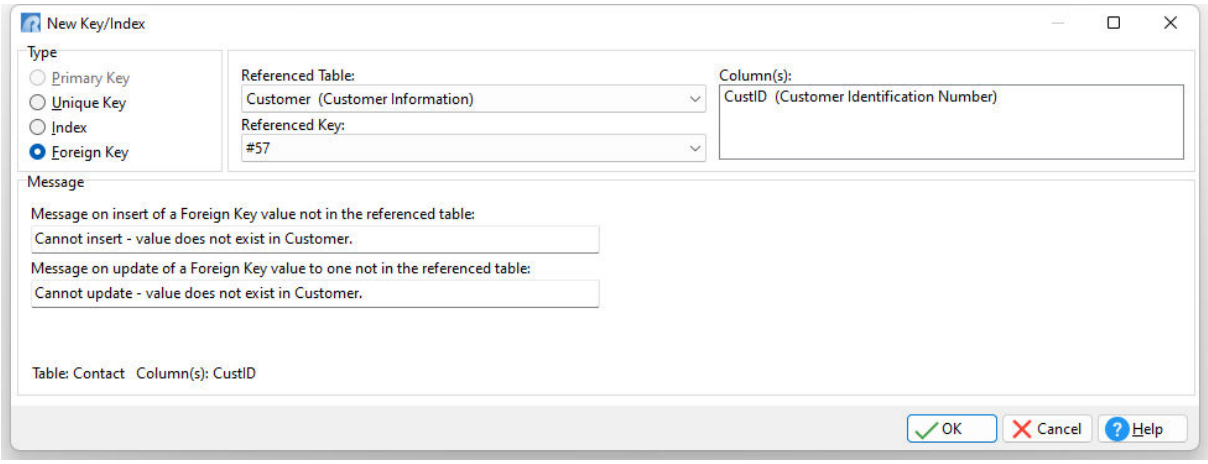


When adding a Unique Key, the options for "Not Null" and "Unique" are disabled by default and cannot be altered. Custom constraint violation error messages can be specified for when an attempt is made to compromise the table's unique key referential integrity.

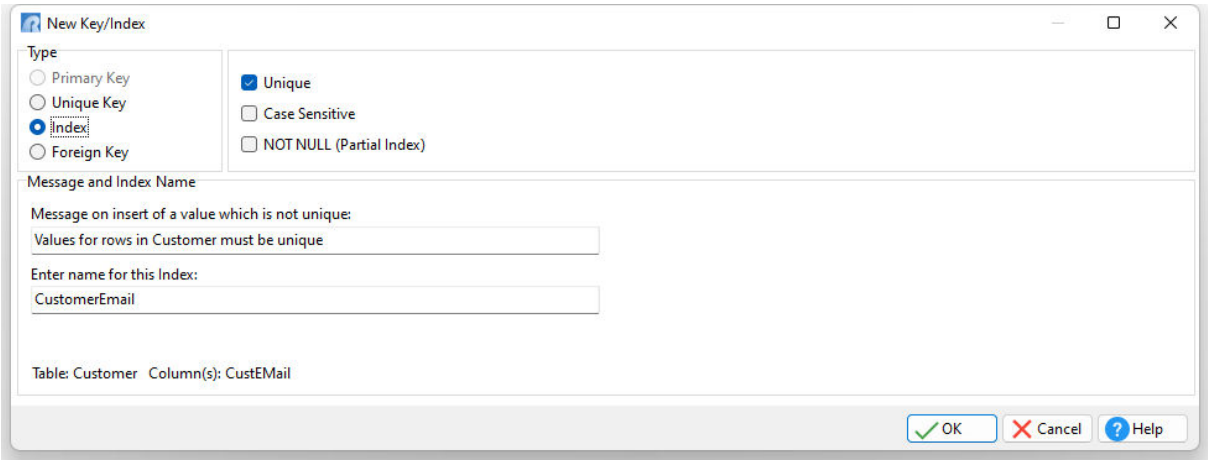


When adding a Foreign Key, the options for to specify the referenced table and primary key is available. Along with a primary key, a unique key cal also be referenced by a foreign key. Custom constraint violation error messages can be specified for when an attempt is made to compromise the table's foreign key referential integrity.

Copyright © 1982-2024 R:BASE Technologies, Inc.



When adding a Index, the options for "Not Null" and "Unique" are optional. By enabling the "Unique"setting, the index will be defined as a unique index. A custom constraint violation error message can be specified for when an attempt is made to enter a non-unique value.



Commands

Defining a primary key or unique key constraint with the CREATE TABLE or ALTER TABLE commands requires the column be explicitly defined as NOT NULL as well as the unique or primary key designation. For example,

```
CREATE TABLE tblname (colname datatype NOT NULL PRIMARY KEY, ...)
```

Below are examples of the ALTER TABLE command to create a primary key for the Employee table, and to create a foreign key for the Employee table that references the Titles table.

```
ALTER TABLE `Employee` ADD PRIMARY KEY +
(`EmpID` ) +
('There must be a unique id number per employee.',+
 'You cannot delete an employee id that is being referenced in another table.',+
 'Cannot change employee id value that is referenced in another table.')

ALTER TABLE `Employee` ADD FOREIGN KEY +
 ( `EmpTID` )+
 REFERENCES `Titles` +
('You must enter an employee id number that exists in the Titles table.',+
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
'You cannot update the title id to this value, use a value that exists in +
the Titles table.')
```

The following example displays the CREATE INDEX command to create a unique index.

```
CREATE UNIQUE INDEX TUII ON `TInvoiceHeader` (`TTransID` ASC )
```

1.31.4 Listing Constraints

The LIST command is used to display information about a database. Additional parameters to display specific constraint information include:

- LIST CONSTRAINTS
- LIST PKEYS
- LIST FKEYS
- LIST UKEYS
- LIST INDEXES
- LIST CASCADE

LIST CONSTRAINTS shows primary key, foreign key, unique key, and Not NULL constraints. The constraint ID, the type of constraint and if it is referenced, the table name, and table references if the key was a foreign key are displayed. Unique indexes are listed next with the index name, table and column names. Not null constraints are listed last with the table and column names.

```
R>LIST CONSTRAINTS

Id     Type                         Table Name         References
------ ---------------------------- -----------------  --------------
#31    PRIMARY KEY REFERENCED       Customer
#33    PRIMARY KEY REFERENCED       Component
#41    FOREIGN KEY                  SalesBonus         Employee
#43    FOREIGN KEY                  CompUsed           Component
#42    FOREIGN KEY                  CompUsed           Product
#44    FOREIGN KEY                  ProdLocation       Product
#45    FOREIGN KEY                  Levels             Product
#34    PRIMARY KEY                  Levels
#35    PRIMARY KEY REFERENCED       Product
#47    FOREIGN KEY                  InvoiceHeader      Customer
#46    FOREIGN KEY                  InvoiceHeader      Employee
#36    PRIMARY KEY REFERENCED       InvoiceHeader
#49    FOREIGN KEY                  InvoiceDetail      Product
#48    FOREIGN KEY                  InvoiceDetail      InvoiceHeader
#51    FOREIGN KEY                  ContactCallNotes   Contact
#50    FOREIGN KEY                  ContactCallNotes   Employee
#52    FOREIGN KEY                  Contact            Customer
#37    PRIMARY KEY REFERENCED       Contact
#38    PRIMARY KEY REFERENCED       Titles
#53    FOREIGN KEY                  Employee           Titles
#39    PRIMARY KEY REFERENCED       Employee
#40    PRIMARY KEY                  StateAbr

Unique Indexes:
Index Name          Table Name           Column Name
------------------ -------------------- --------------------------
UI_Comp            Component            CompDesc


NOT NULL Constraints:
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
Table Name         Column Name
------------------ ------------------
Customer             CustID
Component            CompID
ProdLocation         OnHand
Levels               ModLevel
                     Model
Product              Model
InvoiceHeader        TransID
ContactCallNotes     CallTime
Contact              ContID
Titles               EmpTID
Employee             EmpID
StateAbr             State
```

LIST PKEYS shows only primary keys. The constraint ID, the type of constraint and if it is referenced, and the table name.

```
R>LIST PKEYS

Id     Type                         Table Name         References
------ ---------------------------- ------------------ ------------
#31    PRIMARY KEY REFERENCED       Customer
#33    PRIMARY KEY REFERENCED       Component
#34    PRIMARY KEY                  Levels
#35    PRIMARY KEY REFERENCED       Product
#36    PRIMARY KEY REFERENCED       InvoiceHeader
#37    PRIMARY KEY REFERENCED       Contact
#38    PRIMARY KEY REFERENCED       Titles
#39    PRIMARY KEY REFERENCED       Employee
#40    PRIMARY KEY                  StateAbr
```

LIST FKEYS shows only foreign keys. The constraint ID, the type of constraint, the table name, and table references are displayed. LIST UKEYS displays similar results.

```
R>LIST FKEYS

Id     Type                        Table Name         References
------ --------------------------- ------------------ ---------------
#41    FOREIGN KEY                 SalesBonus           Employee
#43    FOREIGN KEY                 CompUsed             Component
#42    FOREIGN KEY                 CompUsed             Product
#44    FOREIGN KEY                 ProdLocation         Product
#45    FOREIGN KEY                 Levels               Product
#47    FOREIGN KEY                 InvoiceHeader        Customer
#46    FOREIGN KEY                 InvoiceHeader        Employee
#49    FOREIGN KEY                 InvoiceDetail        Product
#48    FOREIGN KEY                 InvoiceDetail        InvoiceHeader
#51    FOREIGN KEY                 ContactCallNotes     Contact
#50    FOREIGN KEY                 ContactCallNotes     Employee
#52    FOREIGN KEY                 Contact              Customer
#53    FOREIGN KEY                 Employee             Titles
```

LIST INDEXES will display all database indexes, and notes if a unique index is defined with (U) to the left of the index name.

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
R>LIST INDEXES

Number of Indexes in Database RRBYW17 is 54.
   Index Name           Table Name          Column Name
   -------------------- ------------------- --------------------
   CustState            Customer            CustState
(U)UI_Comp              Component           CompDesc
```

LIST CASCADE displays tables with CASCADE, and whether UPDATE, DELETE, or BOTH is enabled.

```
R>LIST CASCADE
   Tables with CASCADE flag in the Database RRBYW19

   Name              Cascade Type
   ----------------- --------------------------------
   Customer          CASCADE BOTH
   Employee          CASCADE BOTH
   Titles            CASCADE BOTH
```

The LIST command can also be used to display constraints for a single table. When specifying an individual table, the referenced column names are also displayed.

```
R>LIST CONSTRAINTS FOR Employee

     Table Name: Employee

Id     Type          Column Name(s)  Ref Table Name  Ref Column Name(s)

------ ------------- --------------- --------------- ------------------
#53    FOREIGN KEY   EmpTID          Titles          EmpTID
#39    PRIMARY KEY   EmpID

NOT NULL Constraints:
Table Name         Column Name
------------------ ------------------
Employee           EmpID


R>LIST FKEYS FOR InvoiceHeader

     Table Name: InvoiceHeader

Id     Type           Column Name(s)  Ref Table Name  Ref Column Name(s)

------ -------------- --------------- --------------- ------------------
#47    FOREIGN KEY    CustID          Customer        CustID
#46    FOREIGN KEY    EmpID           Employee        EmpID
```

1.31.5 Removing Constraints

A primary key constraint that is referenced by a foreign key cannot be removed until the foreign key constraint has first been deleted.

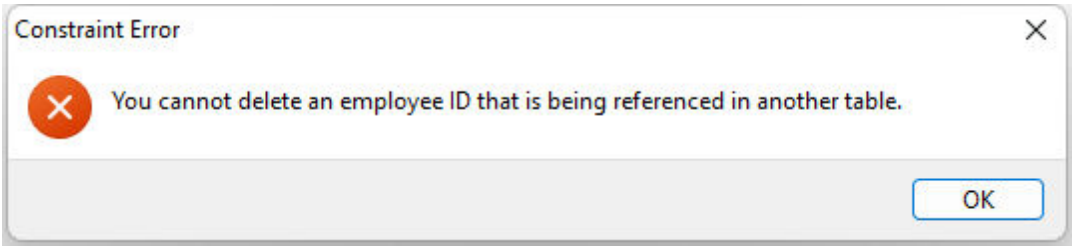
Copyright © 1982-2024 R:BASE Technologies, Inc.

If a column was first defined as not null, and then as a primary key, the not null constraint on the column cannot be removed until the primary key constraint has been removed. Removing a primary key constraint does not remove the NOT NULL part of the constraint. That must be removed separately.

1.31.6 Constraint Messages

When primary key, foreign key and not null constraints are defined, custom violation messages can be entered. The messages cannot be added or modified after the constraint is defined. The constraint must be deleted and re-defined to add or modify custom messages.

The following is an example of a constraint violation error message.

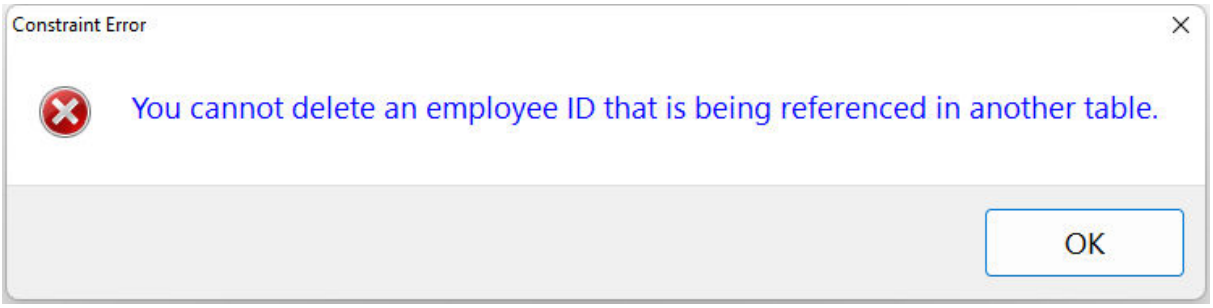


Using two RBTI System Variables, the font size and color can be adjusted for the constraint error message. The variables are:

- RBTI_CEM_FONTSIZE
- RBTI_CEM_FONTCOLOR

Using the following commands, the constraint error message will display a blue font size of 14

```
SET VAR RBTI_CEM_FONTSIZE INTEGER = 14
SET VAR RBTI_CEM_FONTCOLOR TEXT = BLUE
```



1.32 Reserved Words

Do not use reserved words or any shorter forms of them as names for columns, tables, or views. As a rule, if the word is a reserved word, R:BASE will flag it. R:BASE will not allow you to use a reserved word as a column or table name, but this MAY NOT always be the case. For example in the table designer, R:BASE may not warn you about **REF**, short for **REFERENCES**, but **REF** will not be allowed in a command file. If a particular column or table is giving you problems, please check out the list below and consider all shortened versions of the words listed here.

The following is a list of all reserved words.

#DATE	LE
#PI	LEAD
#TIME	LIKE
ADA	LIMIT

Copyright © 1982-2024 R:BASE Technologies, Inc.

ADD	LISTREL
ALL	LT
AND	MAX
ANY	MAXIMUM
AS	MIN
ASC	MINIMUM
ASCII	MODULE
AT	MPW
AUTHORIZE	NE
AUTONUMBER	NOCHECK
AVERAGE	NOFILL
AVG	NONE
BEGINS	NONUM
BETWEEN	NOT
BLINKING	NULL
BOTH	NUM
BY	OF
CASCADE	OFF
CHARACTER	ON
CHECK	OPTION
CMDHIST	OR
COLUMNS	ORDER
CONSTRAINT	OUTER
CONTAINS	OVER
CONTINUE	OWNER
COUNT	PARTITION
CURRENT	PLI
CURSORS	PRECISION
DATA	PRINTER
DBCONN	PRIVILEGE
DECIMAL	PROCEDURE
DESC	PUBLIC
DISTINCT	READ
DUPLICATE	REFERENCES
END	RPW
ENDC	SCHEMA
EQ	SCREEN
ESCAPE	SECTION
EXCEPT	SELECT
EXECUTE	SMALLINT
EXISTS	SOME
EXPLAIN	SORTED
FAILS	SOUNDS_LIKE
FILL	SQLCODE
FLOAT	SQLERROR
FOR	STATICVAR
FOUND	STDEV
FROM	STRUCTURE
FULL	SUM
GE	TABLE
GO	TABLES
GOTO	TEMPORARY
GROUPED	TERMINAL
GT	TJOURNAL
HAVING	TO
IN	UNION
INDEX	UNIQUE
INDICATOR	USING
INTO	VALUES
IS	VARIANCE
KEY	VIEWS
KEYBOARD	WHERE
LAG	WITH
LANGUAGE	WORK
LAST	

Copyright © 1982-2024 R:BASE Technologies, Inc.

Of the above, the following are new reserved words added with the release of R:BASE 11.

- CMDHIST
- STATICVAR
- EXPLAIN
- DBCONN
- TJOURNAL

1.33 SQL - Information

Structured Query Language (SQL) commands provide a standard, machine-independent relational database language.

The American National Standards Institute (ANSI) provides a basic description of the SQL language. The R:BASE/Oterro database provides a superset of this standard. The Oterro database implementation meets the requirements of ANSI 1989 Level 2 SQL with 1992 extensions.

SQL, a language developed specifically for relational databases, enables you to define, modify, and query a relational database. SQL is not a programming language, but a standard set of commands that work with a relational database.

The R:BASE/Oterro database incorporates the SQL commands into its broader range of commands. SQL commands are not treated separately in the Oterro database command because they are not special in any sense. SQL was originally defined as and must be considered an intrinsic part of a database management system.

SQL provides the following sets of commands:

Data Definition Language

Includes the commands needed to create the basic database structures—tables, columns, and views.

- ALTER TABLE (extension to SQL)
- COMMENT ON (extension to SQL)
- CREATE INDEX (extension to SQL)
- CREATE SCHEMA AUTHORIZATION
- CREATE TABLE
- CREATE VIEW
- DROP INDEX (extension to SQL)
- DROP TABLE (extension to SQL)
- DROP VIEW (extension to SQL)

Data Manipulation Language

Provides modification and query capabilities.

- DELETE
- INSERT
- SELECT
- UPDATE

Data Security Language Commands

Control access to the database.

- GRANT
- REVOKE (extension to SQL)

Copyright © 1982-2024 R:BASE Technologies, Inc.

Transaction Processing Commands

Control when data is saved in the database, thereby allowing the restoration of data to a previous state.

- SET AUTOCOMMIT (extension to SQL)
- SET LOCK (extension to SQL)
- SET MAXTRANS (extension to SQL)
- SET TRANSACT (extension to SQL)

1.34 Startup Files

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

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

Startup files can be specified in two ways:

1. **Desktop Shortcut Properties**

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

2. **R:BASE Configuration File**

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

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

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

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

Notes:

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

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

Examples:

Example 01:

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

Example 02:

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

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

1.35 Stored Procedures & Triggers

A stored procedure is a collection of R:BASE commands and/or SQL statements that are stored in the database.

Stored procedures offer a powerful way for developers to add value and ease of maintenance to their R:BASE databases and applications. Moving some of the common business and data access logic out of the R:BASE program into the database centralizes functionality in one place, making it accessible to the R:BASE program as well as any third party ODBC application.

You can run a stored procedure "manually" using the CALL command, or "automatically" by using database triggers.

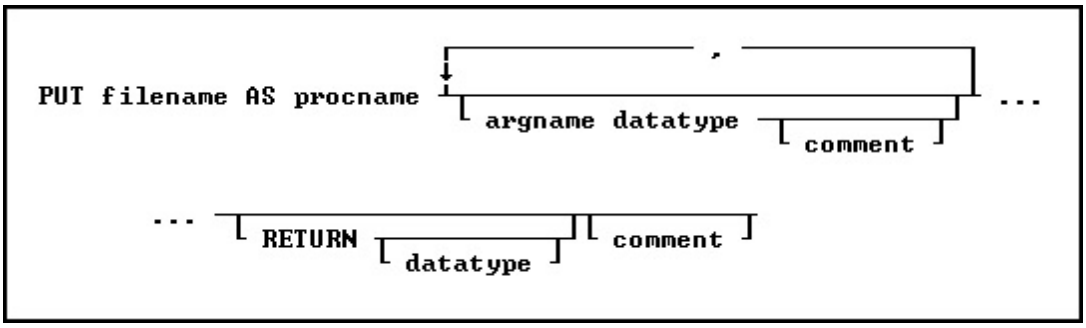
Copyright © 1982-2024 R:BASE Technologies, Inc.

1.35.1 Creating Stored Procedures

Stored procedures are created based upon an existing command file. The process of loading the command file into the database as a stored procedure can be performed through the Database Explorer "Stored Procedure" Group Bar menu option, or by using the PUT command.

PUT

A stored procedure can be created in the R:BASE Editor, and then can be loaded into the database as follows:



Options:

argname datatype
The argument name and data type.

comment
An optional comment for the argument or, if placed after RETURN, an optional comment for the entire procedure. The comment must be enclosed in the current QUOTES character.

filename
The filename in ASCII text format, with full path, to load as the stored procedure.

procname
Specifies the procedure name. If a procedure by this name already exists in the database, an error is generated. The procedure name is limited to 128 characters.

RETURN datatype
Data type of the value returned by the procedure.

Note:

- To clear any previous arguments that were stored for a procedure, use the PUT command as follows:

```
PUT filename AS procname RETURN
```

1.35.2 Using Stored Procedures

Argument List
When you load a stored procedure into a database, you specify arguments to be passed to it. These arguments are used within the procedure. When the procedure is called, the number and type of arguments passed must match the number and type specified when the procedure was stored in the database. When an argument name is referenced in the stored procedure code, the argument name must be preceded by a period unless it is a table or column name, then it must be preceded by an ampersand (&). For example:

```
UPDATE &ProcedureTable SET ColumnName = 100 WHERE ColumnName = .vValue
```

In addition to containing a table name or column name the ampersand variable would be used, if the variable is part of a command, is located on the left side of an operator, contains an ORDER BY clause, or contains a WHERE clause.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Return Values
The value to be returned by a stored procedure is specified in the procedure code following the keyword RETURN. For example, RETURN 'Los Angeles'.

The value returned must match the data type specified when the procedure was stored.

Notes:

- If you are replacing an existing procedure, you must LOCK the procedure first with either the GET LOCK or the SET PROCEDURE command. Once the procedure is locked, it is replaced by an updated file using the PUT command. A procedure cannot be replaced unless it is locked. A procedure is automatically unlocked when replaced with the PUT command.
- The RETURN varname option is used ONLY within a stored procedure to return a value. The returned value is stored in the STP_RETURN system variable. This option will return an -ERROR- when used outside a stored procedure. The default is TEXT 8 characters, but if you want more, you can set it to a larger value. You can control the maximum length at procedure definition time, or by editing the SYS_PROC_LEN column in SYS_PROC_COLS system table.

  To set the limit for the RETURN value to 30 characters:

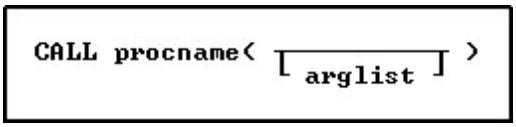
  ```
  PUT MyTest.PRC AS MyTest P1 INTEGER RETURN TEXT (30)
  ```

- To clear any previous arguments that were stored for a procedure:

  ```
  PUT filename AS procname RETURN
  ```

1.35.2.1 CALL

To run a stored procedure you can use the CALL command like a function or run the CALL command at the R> Prompt or in a command file.



Use the CALL command like a function with the following syntax:

```
SET VAR v1 = (CALL procname(arglist))
```

Run the CALL command from the R> Prompt with the following syntax:

```
CALL procname(arglist)
```

When the CALL command is run at the R> Prompt, the return value from the stored procedure is placed in the variable STP_RETURN. The return value can be an expression.

procname
The procedure name.

arglist
The argument values separated by commas. An *arglist* must always be used, even if empty. For example:

```
SET VAR v1 = (CALL procname ( )).
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.35.2.2 GET

The GET command is used to read a stored procedure from the database into an ASCII command file. If the LOCK option is used with the GET command, the procedure can be replaced by using PUT.

Options:

filename
The name of the ASCII text format file the stored procedure is placed in.

LOCK
Locks the procedure so it cannot be locked or unlocked by another user. When a procedure is locked, only the user placing the lock can replace the procedure. The NAME setting is used for identification of the user.

procname
Specifies the procedure name. The procedure name is limited to 128 characters.

1.35.2.3 SET PROCEDURE

The SET PROCEDURE command locks a procedure so it can be replaced. It works like the GET LOCK command without retrieving the stored procedure into an ASCII file. ON sets the lock; OFF releases the lock placed by the SET PROCEDURE or the GET commands.

When a procedure is locked, only the user placing the lock can replace the procedure or remove the lock. The NAME setting is used for identification of the user.

1.35.2.4 Examples

To Create a Stored Procedure

Create the following command file, INS.RMD, in R:BASE Editor:

```
*(INS.RMD:)
IF (.p1 > 105) THEN
  INSERT INTO contact (custid, contlname) VALUES (.p1, .p2)
  RETURN 1
ELSE
  RETURN 0
ENDIF
```

To create the stored procedure from the .RMD file:

```
PUT INS.RMD AS proc1 p1 INT, p2 TEXT RETURN INTEGER
```

The following stored procedure example will generate one new row in *contact* and set v1 = 1.

```
SET VAR vname = 'Dunn'
SET VAR v1 = (CALL proc1 (106, .vname))
```

The following stored procedure example will set v1 = 0.

```
SET VAR vname = 'Dunn'
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
SET VAR v1 = (CALL proc1 (100, .vname))
```

To Delete a Stored Procedure

To delete a stored procedure, use the DROP command with the following syntax:

```
DROP PROCEDURE procname
```

Optionally, you can enter the following code at the R> Prompt:

```
SET VAR vProcID = sys_proc_id IN sys_procedures +
  WHERE sys_proc_name = 'procname'
DELETE ROWS FROM sys_procedures +
  WHERE sys_proc_id = .vProcID
DELETE ROWS FROM sys_proc_mods +
  WHERE sys_proc_id = .vProcID
DELETE ROWS FROM sys_proc_cols +
  WHERE sys_proc_id = .vProcID
```

To Rename a Stored Procedure

To rename a stored procedure, use the RENAME command with the following syntax:

```
RENAME PROCEDURE procname1 TO procname2
```

To List Stored Procedures

With the LIST command, you can list every stored procedure in a database or list information about a specific stored procedure.

To display the name and a description for every procedure in the open database, use the following syntax:

```
LIST PROCEDURE
```

To display a specific procedure and its attributes, use the following syntax:

```
LIST PROCEDURE procname
```

This option displays the name, description, ID, date last modified, version, locked by (if locked), and if applicable, the return type and description for the specified stored procedure. If the stored procedure has arguments, the number of arguments and argument names and attributes will be listed.

To Trace a Stored Procedure

With the TRACE command, you can debug the stored procedure command syntax in the R:BASE Trace Debugger.

Enter the following code at the R> Prompt:

```
TRACE SELECT USING .arg1, .arg2 SELECT sys_proc_src FROM sys_procedures +
  WHERE sys_proc_name = 'procname'
```

1.35.3 Restricted Commands

Stored procedures can contain all R:BASE/SQL commands except for the following:

CODELOCK	FORMS
RBEDIT	RESTORE
CONNECT	PACK
RBLABELS	RULES

Copyright © 1982-2024 R:BASE Technologies, Inc.

DISCONNECT	SET (without a keyword)
RBAPP	RBDEFINE
REPORTS	TRACE
GRANT	REVOKE

1.35.4 Stored Procedure System Tables

The R:BASE system tables are created by R:BASE when a database is created. They contain system information. The following are new system tables. Stored procedures are stored in the database in the system table called SYS_PROCEDURES. Supporting system tables are SYS_PROC_COLS and SYS_PROC_MODS.

Table: sys_PROCEDURES

Column Name	Data Type	Description
sys_proc_id	INTEGER	Procedure identification
sys_proc_name	NOTE	Procedure name
sys_proc_locked_by	NOTE	Last user to do a PUT or LOCK
sys_proc_comment	NOTE	Descriptive comment for procedure
sys_proc_src	LONG VARCHAR	Procedure source
sys_proc_mod_ts	DATETIME	Timestamp of procedure
sys_proc_obj	LONG VARBIT	Reserved for future use
sys_proc_usage	INTEGER	Reserved for future use
sys_proc_flags	INTEGER	Internal binary flags. Bit 0 is the LOCK flag.
sys_proc_version	INTEGER	Version number of procedure

Table: sys_proc_cols

Column Name	Data Type	Description
sys_proc_col_id	INTEGER	Argument identification
sys_proc_id	INTEGER	Procedure identification
sys_proc_col_name	NOTE	Name of argument
sys_proc_iotype	INTEGER	Argument type values equal to: SQL_RETURN_VALUE (5) SQL_PARAM_OUTPUT (4) SQL_PARM_INPUT (1) SQL_PARAM_INPUT_OUTPUT (2) Currently only INPUT and RETURN types are supported.
sys_proc_datatype	INTEGER	Data type of argument
sys_proc_len	INTEGER	Argument data length
sys_proc_scale	INTEGER	Argument data scale
sys_proc_flags	INTEGER	Internal binary flags
sys_proc_comment	NOTE	Descriptive comment for argument
sys_proc_defvalu	NOTE	Reserved for future use

Table: sys_proc_mods

Column Name	Data Type	Description
sys_proc_mod_id	INTEGER	Archive identification
sys_proc_id	INTEGER	Procedure identification
sys_proc_mod_ts	DATETIME	Timestamp of archived procedure
sys_proc_user	NOTE	User who did a PUT on procedure
sys_proc_comment	NOTE	Descriptive comment for procedure
sys_proc_fc	LONG VARCHAR	Reserved for future use
sys_proc_delta	LONG VARBIT	Reserved for future use
sys_proc_version	INTEGER	Version of archive

Copyright © 1982-2024 R:BASE Technologies, Inc.

Table: sys_TRIGGERS

Column Name	Data Type	Description
sys_table_id	INTEGER	Table identification
sys_trig_ins	INTEGER	Id of INSERT procedure
sys_trig_upd	INTEGER	Id of UPDATE procedure
sys_trig_del	INTEGER	Id of DELETE procedure

1.35.5 Triggers

A database trigger is procedural code that is automatically executed in response to certain events on a particular table in the database. Triggers can restrict access to specific data, perform logging, or audit data modifications.

There are "BEFORE" triggers and "AFTER" triggers which identifies the time of execution of the trigger. A trigger can be set to automatically run a stored procedure before and/or after an update, delete, or insert event occurs in a table.

There are three triggering events that cause triggers to fire:

- INSERT event (as a new record is being inserted into the database).
- UPDATE event (as a record is being changed).
- DELETE event (as a record is being deleted).

Notes:

- R:BASE triggers occur only once per INSERT, UPDATE, or DELETE event.
- A table can have both "BEFORE" and "AFTER" triggers, only one, or none.
- Triggers are stored in the System Table SYS_TRIGGERS
- Procedures that are run with triggers must be stored with no arguments. See "**Argument List**" under Using Stored Procedures

1.35.5.1 Using Triggers

The update, delete, or insert event can be initiated through the UPDATE, DELETE, or INSERT R:BASE/SQL commands, or through an R:BASE form.

Typical Trigger Usage:

- BEFORE - data validation before the action (inventory checks, account limit checks)
- AFTER- update of data after the action (dependent on primary keys, automated post transaction steps)

Since a "BEFORE" trigger runs a stored procedure before the row that triggered it is updated, inserted, or deleted, you can cancel the update, insert, or delete with the ABORT TRIGGER command in the stored procedure. Since the modified data has been "committed" with an "AFTER" trigger, you cannot abort the action in the stored procedure.

Also, you can verify the action being performed in an update trigger on the row by using a SELECT command with the WHERE CURRENT OF SYS_OLD/SYS_NEW syntax to check the row before/after the update.

Creating a Trigger

Triggers can be created using the Data Designer, or with the CREATE TABLE or ALTER TABLE commands. When you use the ALTER TABLE command you must define the insert triggers in the same command. The same applies for update and delete. Do not use one alter table command to add the "BEFORE" insert trigger and then another alter table to add the "AFTER" trigger. Do them both in the same command.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Removing a Trigger

When you drop a trigger with the ALTER TABLE command, you do not have to specify the "BEFORE" or "AFTER" trigger. The DROP of the insert trigger, for example, drops both parts if they exist.

Listing Defined Triggers

To LIST all the tables in the open database that have triggers and the triggers, use the following syntax:

```
LIST TRIGGER
```

To list triggers for a specified table, use the following syntax:

```
LIST TRIGGER tblname
```

Example:

```
ALTER TABLE TableName ADD TRIGGER INSERT ProcName
ALTER TABLE TableName ADD TRIGGER UPDATE ProcName
ALTER TABLE TableName ADD TRIGGER DELETE ProcName
ALTER TABLE TableName ADD TRIGGER INSERT AFTER ProcName
ALTER TABLE TableName ADD TRIGGER UPDATE AFTER ProcName
ALTER TABLE TableName ADD TRIGGER DELETE AFTER ProcName
```

See also:

SYS_NEW
SYS_OLD

For a sample database using triggers, please locate the "Stored Procedures, Triggers and After Triggers" sample located at http://www.razzak.com/sampleapplications/

1.35.5.2 SYS_NEW

The SYS_NEW parameter is used in a WHERE clause within, and only within, the context of a trigger.

This virtual pointer is available to INSERT and UPDATE triggers. It allows code to access the contents of the row as it will be after the INSERT or UPDATE action. Using this in the body of a WHERE clause allows code to act on the contents of that virtual row and NOT fire off another trigger.

The following is a list the trigger types available for use with SYS_NEW, and whether or not they are updatable:

- BEFORE INSERT: Updatable
- AFTER INSERT: Read only
- BEFORE UPDATE: Updatable
- AFTER UPDATE: Read only

Note: The use of functions or expressions must be performed outside of the virtual pointer SELECT statement, after the the variable values are captured.

Example

The following command is within the body of an Insert Trigger and is being used to increment the count of how many items have been used.

```
SELECT ProductType INTO vPType INDIC v1 +
  FROM SalesDetails +
  WHERE CURRENT OF SYS_NEW
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
UPDATE ProductCount +
  SET ProductCount = (ProductCount + 1) +
  WHERE ProductType = .vPType
```

1.35.5.3 SYS_OLD

The SYS_OLD parameter is used in a WHERE clause within, and only within, the context of a trigger.

This virtual pointer is available to UPDATE and DELETE triggers. It allows code to access the contents of the row as it will be prior the UPDATE or DELETE action. Using this in the body of a WHERE clause allows code to act on the contents of that virtual row and NOT fire off another trigger.

The following is a list the trigger types available for use with SYS_OLD, and whether or not they are updatable:

- BEFORE DELETE: Read only
- BEFORE UPDATE: Read only

Note: The use of functions or expressions must be performed outside of the virtual pointer SELECT statement, after the the variable values are captured.

Example

The following command is within the body of an Delete Trigger and is being used to automatically archive a message from an employee messaging table.

```
SELECT EmployeeID,MsgDate,MsgTime,MsgBody +
  INTO vEID INDIC v1,vMsgDate INDIC v2, +
  vMsgTime INDIC v3,vMsgBody INDIC v4 +
  FROM EmpMessage WHERE CURRENT OF SYS_OLD
INSERT INTO ArchMessage +
  (EmployeeID,MsgDate,MsgTime,MsgBody,DeletedOn) +
  VALUES .vEID,.vMsgDate,.vMsgTime,.vMsgBody,.#DATE
```

1.36 System Variables

R:BASE presets certain variables, which always exist while the program is running. These variables are described in the following table:

Variable Name	Use	Example of Use
#DATE	Stores the current system date	`SET VAR vToday = .#DATE`
#TIME	Stores the current system time	`SET VAR vTimer = (.#TIME + 10)`
#PI	Stores the value of as a DOUBLE data type (3.14159265358979)	`SET VAR vNum = (.#PI *.rad**2)`

SQLCODE	Holds the result of the previous SQL command	`SELECT netamount FROM transmaster +`
		`  WHERE netamount IS NULL`
		`IF SQLCODE <> 100 THEN`
		`  --Perform Task here`
		`ENDIF`
		
		The IF...ENDIF condition checks if any rows exist. If valid rows are found, SQLCODE is set to 0, and the control passes to the command after ENDIF. If no data is found, SQLCODE is set to 100. A negative number

Copyright © 1982-2024 R:BASE Technologies, Inc.

		for SQLCODE indicates failure. Based on the value of *SQLCODE*, the WHENEVER command can be used to call an error-handling routine.
SQLSTATE	Stores a 5-character long return code string that indicates the status of the previous SQL statement. SQLSTATE was added for Oterro compatibility. Values may be set by attached server tables commands or stored procedures. A list of valid SQLState values is part of the Oterro documentation.	
#NOW	Stores the current system date and time	`SET VAR vRightNow DATETME = .#NOW`
RBTI_PRNSETUP	Stores the user's button selection after using the PRNSETUP dialog	
RBTI_CEM_FONTSIZE	Can be declared to alter the font size for constraint error messages	`SET VAR RBTI_CEM_FONTSIZE INTEGER = 14`
RBTI_CEM_FONTCOLOR	Can be declared to alter the font color for constraint error messages	`SET VAR RBTI_CEM_FONTCOLOR TEXT = 'BLUE'`
RBTI_ERR_FONTSIZE	Can be declared to alter the font size for general error messages	`SET VAR RBTI_ERR_FONTSIZE INTEGER = 12`
RBTI_ERR_FONTCOLOR	Can be declared to alter the font color for general error messages	`SET VAR RBTI_ERR_FONTCOLOR TEXT = 'RED'`
RBTI_IMG_FOLDER	Can be declared to define a folder path where all "Open Image" dialogs will initially start. If the RBTI_IMG_FOLDER variable does not exist or contains a non-valid folder path, all open image dialogs will launch within the current folder.	`SET VAR RBTI_IMG_FOLDER TEXT = 'C:\TEMP\IMAGES\'`
RBTI_EXPORT_FOLDER	Can be declared to define a folder path where Data Browser "Export" dialogs will initially start. If the RBTI_EXPORT_FOLDER variable does not exist or contains a non-valid folder path, all open image dialogs will launch within the current folder.	`SET VAR RBTI_EXPORT_FOLDER TEXT = 'D:\EXPORTS\'`
RBTI_PRNDEFDIR	Can be declared to define the default directory for the "Print to File" print dialog, before the dialog is launched	`SET VAR RBTI_PRNDEFDIR TEXT = 'C:\Temp\'`
RBTI_PRNDEFFILE	Can be declared to define the default file name for the "Print to File" print dialog, before the dialog is launched	`SET VAR RBTI_PRNDEFFILE TEXT = 'CSI_Log'`
RBTI_PRNDEFFILET	Can be declared to define the default output type for the "Print to File" print dialog, before the dialog is launched	`SET VAR RBTI_PRNDEFFILET TEXT = 'PDF File'`

Copyright © 1982-2024 R:BASE Technologies, Inc.

With the FEEDBACK operating condition set ON to display processing results, the below commands will create FEEDBACK system variables containing values.

1. ALTER TABLE
2. COMPUTE
3. CREATE INDEX
4. CROSSTAB
5. DELETE
6. INSERT
7. The data transfer for the JOIN of two tables
8. LOAD
9. The data transfer for a PROJECT command
10. SELECT
11. SORTing a large record set
12. The data transfer for a SUBTRACT of two tables
13. TALLY
14. The UNION command

The FEEDBACK system variables will be generated and hold the values for the record changes and the elapsed time for the command.

Variable Name	Use
RBTI_RowsInserted	Holds the value for the number of rows inserted
RBTI_RowsDeleted	Holds the value for the number of rows deleted
RBTI_RowsUpdated	Holds the value for the number of rows updated
RBTI_ElapsedTime	Holds the value for the elapsed processing time

You can verify these values by typing SHOW VARIABLES at the R> Prompt.

1.36.1 Forms

R:BASE supports the following RBTI System Variables used in Forms:

- **RBTI_DBGRID_COLUMN**
  This variable holds the name of focused column for a DB Grid.
- **RBTI_DIRTY_FLAG**
  Returns 1 if any DB Control value(s) in form were changed or 0 if nothing was changed.
- **RBTI_FORM_ALIAS**
  This variable holds the name of focused form, if used AS alias.
- **RBTI_FORM_CLICKED**
  This variable is set to the clicked control's Component ID property (if exists). The value can then be used by the PROPERTY/GETPROPERTY commands to manipulate the control.
- **RBTI_FORM_COLNAME**
  This variable holds the name of focused column DB Control on form.
- **RBTI_FORM_COLVALUE**
  This variable holds the value of focused column DB Control on form.
- **RBTI_FORM_COMPID**
  This variable holds the value of focused DB/Variable Edit control's Component ID, if defined.
- **RBTI_FORM_CTRLTEXT**
  This variable is set to the clicked control's "Caption" or "Text" property. Display-only controls like a Label and Panel use the "Caption" property to hold text. For entry controls like DB/Variable Edit or Combo Boxes, the property to hold text is "Text". When setting the value of RBTI_FORM_CTRLTEXT, the control is first checked if it has the "Caption" property. If "Caption" was not found "Text" is used. If both "Caption" and "Text" were not found, the variable is set to null.

Copyright © 1982-2024 R:BASE Technologies, Inc.

- **RBTI_FORM_DATATYPE**
  This variable holds the data type of focused column DB Control on form.

- **RBTI_FORM_DBLCLKED**
  This variable is set to the double clicked control's Component ID property (if exists). The value can then be used by the PROPERTY/GETPROPERTY commands to manipulate the control.

- **RBTI_FORM_DIRTYVAR**
  Returns 1 if any Variable Control value(s) in form were changed or 0 if nothing was changed.

- **RBTI_FORM_FORMNAME**
  This variable holds the name of the current form in use, which is particularly useful when using a form on top of another form or if multiple forms are running at the same time. The variable also provides the current form name displayed in the Form Designer.

- **RBTI_FORM_MODE**
  This variable holds the value of current mode of the form, such as ENTER, EDIT or BROWSE.

  When the form is brought-up as ENTER USING formname ... the value for RBTI_FORM_MODE will be returned as 'ENTER'.

  When the form is brought-up as EDIT USING formname ... the value for RBTI_FORM_MODE will be returned as 'EDIT'.

  When the form is brought-up as BROWSE USING formname ... the value for RBTI_FORM_MODE will be returned as 'BROWSE'.

- **RBTI_FORM_TBLNAME**
  This variable holds the name of the current table in a form. This is especially useful when used within a multi-table form.

- **RBTI_FORM_VARNAME**
  This variable holds the name of focused Variable Control on form.

- **RBTI_FORM_VARVALUE**
  This variable holds the value of focused Variable Control on form.

- **RBTI_FORM_DSGN**
  This variable holds the value of "YES" when a form is open in the Form Designer.

- **RBTI_FORM_EXT_DSGN**
  This variable holds the value of "YES" when a form is open in the External Form Designer.

Tips and Techniques:

- If you have variables controls on a form in EDIT mode and the value of the variable control is changed - then RBTI_FORM_DIRTYVAR will change. You can then use this to take appropriate action.

- In form EDIT mode, RBTI_DIRTY_FLAG will only change if a DB control is changed that affects the displayed dataset. In ENTER mode RBTI_DIRTY_FLAG is always set to 1. In EDIT mode this allows you to easily cater for a form with a mix of DB controls and variable controls - by using RBTI_FORM_DIRTYVAR you can trap in an EEP the situation where the user has only made a change to a variable control and not to any DB controls. (In this situation RBTI_DIRTY_FLAG will be 0 but RBTI_FORM_DIRTYVAR will be 1.)

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

```
PROPERTY APPLICATION CAPTURE_RBTI_FORM_VARS OFF
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.36.2 Reports

R:BASE supports the following RBTI System Variables used in Reports:

- **RBTI_REPORT_NAME**
  This variable holds the name of the current report.
- **RBTI_REPORT_NAME_X**
  This variable captures the name when an attempt is made to open a report/label, whether the end result is successful or not/canceled.
- **RBTI_RSTAT_TIME**
  This variable captures the generation time, in seconds, for the report to print.
- **RBTI_RSTAT_PAGES**
  This variable captures the number of pages for the printed report.
- **RBTI_RSTAT_DEV**
  This variable captures the output device used (SCREEN, PRINTER, FILE) for the print routine.

1.36.3 Labels

R:BASE supports the following RBTI System Variables used in Labels:

- **RBTI_LABEL_NAME**
  This variable holds the name of the current label.

1.36.4 Queries

R:BASE supports the following RBTI System Variables used with views/queries:

- **RBTI_QBE_NAME**
  This variable holds the name of the view used in the Query Builder. The RBTI_QBE_NAME variable will be defined with the view name after any "Save" menu options are used to create a new object, and will also store a view name after the query is opened and closed within the Query Builder.

1.37 Table Joins

A join is an SQL clause that combines records from two tables in a database. When you perform a join, you specify one column from each table to join on. These two columns contain data that is shared across both tables. You can use multiple joins in the same SQL statement to query data from as many tables as you like.

A join can be an inner join, an outer join, or a self join. An inner join, like the INTERSECT command, includes only those rows that match on the linking columns. An outer join, like UNION, includes all rows that match as well as all rows that don't match on linking columns.

Most of the time, you'll do an inner join, though you will sometimes find it useful to do an outer join. For example, you need an outer join (like the UNION command) to get all rows in these cases:

- When joining a *customer* table with an *orders* table to list the customers who ordered something in the current month (inner join) as well as those who didn't order anything (outer join).
- When joining a *budget* table with an *expense* table to list each budget item, whether or not there was an expense for that item in the current month.
- When comparing a header (master table) on the "one" side of a one-to-many relationship against a detail (transaction table) on the "many" side to see all the rows of data, whether or not they have associated details.

In R:BASE, there are two different syntactical ways to express joins. The first, called *explicit join notation*, uses the keyword JOIN, whereas the second is the *implicit join notation*. The implicit join notation uses commas to separate the tables to be joined in the FROM clause of a SELECT statement. Thus, it always computes a cross join, which results in the number of rows in the first table multiplied by

Copyright © 1982-2024 R:BASE Technologies, Inc.

the number of rows in the second table, where a WHERE clause may apply additional filtered criteria. That filter criteria is comparable to join predicates in the explicit notation.

Example of an explicit "inner" join:

```
SELECT ALL FROM Product +
  INNER JOIN TransDetail ON +
  TransDetail.Model = Product.Model
```

Example of an implicit "inner" join:

```
SELECT ALL FROM Product t3, TransDetail t2 +
  WHERE t3.Model = t2.Model
```

Both of the above examples will result in the same output, only the example of the explicit "inner" join will be faster.

1.37.1 Join Types

Depending on your requirements, you can do an "INNER" join or an "OUTER" join. The differences are:

- **INNER JOIN:** This will only return rows when there is at least one row in both tables that match the join condition.
- **LEFT OUTER JOIN:** This will return rows that have data in the left table (left of the JOIN keyword), even if there's no matching rows in the right table.
- **RIGHT OUTER JOIN:** This will return rows that have data in the right table (right of the JOIN keyword), even if there's no matching rows in the left table.
- **FULL OUTER JOIN:** This will return all rows, as long as there's matching data in one of the tables.

1.37.2 More About OUTER JOIN

When you use an outer join, rows are not required to have matching values. The table order in the FROM clause specifies the left and right table. You can include a WHERE clause and other SELECT clause options such as GROUP BY. The result set is built from the following criteria:

- In all types of outer joins, if the same values for the linking columns are found in each table, R:BASE joins the two rows.
- For a left outer join, R:BASE uses each value unique to the left (first) table and completes it with nulls for the columns of the right (second) table when the linking columns do not match.
- A right outer join uses unique values found in the right (second) table and completes the rows with nulls for columns of the left (first) table when the linking columns do not match.
- A full outer join first joins the linking values, followed by a left and right outer join.

Four Examples of Outer Joins

Below, from slowest to fastest, are four examples of how to list all the invoice numbers in an *invoice* table, whether or not they have related rows in a *transx* table.

In each example, *invoice* has 1,000 rows and *transx* has 8,000 rows. There are 20 matches and 980 non-matches. In other words, in the first three examples the first SELECT (inner join) finds 20 rows and the second SELECT (outer join) finds 980 rows. And, in the last example, the LEFT OUTER JOIN performs the query with just one SELECT.

Uncorrelated Sub-SELECT
|
This first example shows how to list the invoice numbers with a simple sub-SELECT that doesn't have a WHERE clause correlating it to the main SELECT. It's slow, taking a long time to complete.

```
SELECT t2.InvId, SUM(t3.TPrice) +
  FROM Invoice t2, Transx t3 +
  WHERE t3.InvId = t2.InvId +
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
  GROUP BY t2.InvId +
  UNION +
    SELECT Invoice.InvId, $0.00 +
      FROM Invoice +
      WHERE InvId NOT IN +
        (SELECT InvId FROM Transx)
```

Correlated Sub-SELECT

Adding a correlated WHERE clause to the sub-SELECT makes it many times faster.

```
SELECT t2.InvId, SUM(t3.TPrice) +
  FROM Invoice t2, Transx t3 +
  WHERE t3.InvId = t2.InvId +
  GROUP BY t2.InvId +
  UNION +
    SELECT t1.InvId, $0.00 +
      FROM Invoice t1 +
      WHERE InvId NOT IN +
        (SELECT InvId FROM Transx +
        WHERE Transx.InvId = t1.InvId)
```

Correlated and NOT EXISTS

By changing NOT IN to NOT EXISTS for use with the correlated sub-SELECT, you can add a little more speed.

```
SELECT t2.InvId, SUM(t3.TPrice) +
  FROM Invoice t2, Transx t3 +
  WHERE t3.InvId = t2.InvId +
  GROUP BY t2.InvId +
  UNION +
    SELECT tl.InvId, $0.00 +
      FROM Invoice t1 +
      WHERE NOT EXISTS +
        (SELECT InvId FROM Transx +
        WHERE Transx.InvId = t1.InvId)
```

LEFT OUTER JOIN

By using a LEFT OUTER JOIN, and bypassing the second SELECT, you can add even more speed.

```
SELECT t2.InvId, SUM(t3.TPrice) +
  FROM Invoice t2 LEFT OUTER JOIN Transx t3 ON +
  t3.InvId = t2.InvId +
  GROUP BY t2.InvId
```

1.38 Temporary Tables and Views

Temporary tables and views are non-permanent tables/views that only exist for the duration of a database session. When a database session terminates, its temporary tables/views are automatically destroyed. Temporary tables/views are only visible to the R:BASE session that creates them and remain invisible to other R:BASE users. Several users can create temporary tables/views with the same name, and each user will see only that particular version of the table/view.

Temporary tables are ideal for holding short-term data used by the current R:BASE session. For example, suppose you need to do many SELECTs on the result of a complex query. An efficient strategy

Copyright © 1982-2024 R:BASE Technologies, Inc.

is to execute the complex query once, then store the result in a temporary table. You can also create an index on the temporary table to speed up queries with the CREATE INDEX command. In addition to indexes, you can create rules, constraints and triggers on temporary tables.

You may CREATE or turn a permanent table into a TEMPORARY table using the enhanced Data Designer.

1.38.1 Using Temporary Tables/Views

Creating Temporary Tables/Views
Use the CREATE TEMPORARY TABLE command to create a temporary table. Use the CREATE TEMPORARY VIEW command to create a temporary view. You can also use the PROJECT command to create a temporary table based upon another table. The main difference between using the original command to create a table or view and creating a temporary table/view is the TEMPORARY parameter in the command syntax.

Temporary Table Setting
Within the R:BASE Data Designer, there is a setting, under "Table", that will save any temporary table as a regular permanent table. The setting can also be used to save any existing table as a temporary table.

Removing Temporary Tables/Views
You can use the DROP command to remove a temporary table or view. This is important when you are creating temporary tables or views as you should always DROP the table or view before creating it. This technically only affects the user running the program. To avoid this error message, use the following technique:

Example 01:
```
SET ERROR MESSAGE 2038 OFF
DROP TABLE temptablename
SET ERROR MESSAGE 2038 ON
```

Example 02:
```
SET ERROR MESSAGE 677 OFF
DROP VIEW tempviewname
SET ERROR MESSAGE 677 ON
```

In addition to the DROP command, you can simply DISCONNECT and CONNECT the database to remove any temporary tables/views. Temporary tables/views along with temporary System Tables are removed when the database is disconnected.

Database Maintenance with Temporary Tables/Views
One should always DISCONNECT and then CONNECT the database before using the RELOAD or UNLOAD commands.

Using Forms, Labels or Reports based upon Temporary Tables/Views
You may define all required TEMPORARY tables/views for Forms as "On Before Design Action" before designing a form and "On Before Start" EEP before using (EDIT USING, ENTER USING, BROWSE USING) the form.

You may define all required TEMPORARY tables/views for Reports as "On Before Design ..." action before designing a report/sub-report and "On Before Generate ..." action.

You may define all required TEMPORARY tables/views for Labels as "On Before Design ..." action before designing a label and "On Before Generate ..." action.

A command file, such as TempTables.RMD, can also be used to pre-define all required temporary tables/views.

Notes:

- In regards to temporary tables in the database, always DISCONNECT and then CONNECT before using the AUTOCHK, PACK or RELOAD commands.
- When creating a temporary table/view, the following temporary System Tables are also created:

Copyright © 1982-2024 R:BASE Technologies, Inc.

- SYSTMP_COMMENTS
- SYSTMP_CONSTRAINTS
- SYSTMP_DEFAULTS
- SYSTMP_RULES
- SYSTMP_SERVERS
- SYSTMP_TRIGGERS
- SYSTMP_VIEWS

- Temporary tables/views were first introduced in R:BASE 6.1 (July 1997).

1.38.2 Differentiate between Regular and Temporary Tables/Views

There are two ways to differentiate between regular and temporary tables/views. One way is to view the Tables or Views menu within the Database Explorer. You will notice that any temporary tables or views will have a faded icon next to the appropriate table/view name.

Another way is using the LIST command at the R> Prompt. To safely indicate which tables are temporary tables in the LIST or LIST TABLES command, you will now see a "(T)" in front of the table name.

Here's how using the ConComp sample database:

1. Launch R:BASE
2. CONNECT ConComp
3. Switch to the R> Prompt and create temporary tables using the following:

```
PROJECT TEMPORARY tCustomer FROM Customer USING ALL
```

Or use the following block of code in a command file to create two temporary tables

```
SET ERROR MESSAGE 2038 OFF
DROP TABLE tInvoiceHeader
CREATE TEMPORARY TABLE `tInvoiceHeader` +
(`TransID` INTEGER, +
`CustID` INTEGER, +
`EmpID` INTEGER, +
`TransDate` DATE, +
`BillToCompany` TEXT (40), +
`BillToAddress` TEXT (30), +
`BillToCity` TEXT (20), +
`BillToState` TEXT (2), +
`BillToZip` TEXT (10), +
`ShipToCompany` TEXT (40), +
`ShipToAddress` TEXT (30), +
`ShipToCity` TEXT (20), +
`ShipToState` TEXT (2), +
`ShipToZip` TEXT (10), +
`NetAmount` CURRENCY, +
`Freight`= ( netamount* .01) CURRENCY, +
`Tax`= ( netamount* .081) CURRENCY, +
`InvoiceTotal`= (NetAmount+Freight+Tax) CURRENCY)
COMMENT ON TABLE tInvoiceHeader IS 'Invoice Header Information'
DROP TABLE tInvoiceDetail
CREATE TEMPORARY TABLE `tInvoiceDetail` +
(`TransID` INTEGER, +
`DetailNum` INTEGER, +
`Model` TEXT (6), +
`Units` INTEGER,  +
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
`Price` CURRENCY, +
`Discount` REAL, +
`SalePrice`= (Price-(Price*Discount/100)) CURRENCY, +
`ExtPrice`= (Units* SalePrice) CURRENCY)
AUTONUM `DetailNum` IN `tInvoiceDetail` USING 1,1
COMMENT ON TABLE tInvoiceDetail IS 'Invoice Header Information'
SET ERROR MESSAGE 2038 ON

LIST TABLES
```

You will notice the (T) in front of the temporary tables.

4. Now create a temporary view using following:

```
SET ERROR MESSAGE 677 OFF
DROP VIEW tYTDInvoiceTotal
CREATE TEMP VIEW `tYTDInvoiceTotal` +
(CustID, YTDInvoiceTotal) AS  +
SELECT CustID,(SUM(InvoiceTotal)) FROM TransMaster +
GROUP BY CustID
COMMENT ON VIEW `tYTDInvoiceTotal` IS +
'Year-To-Date Invoice Total by Customer'
SET ERROR MESSAGE 677 ON

LIST VIEWS
```

You will notice (T) in front of the tYTDInvoiceTotal view.

1.38.3 Advantages of Temporary Tables/Views

Raw Speed

Temporary Tables/Views are lightening fast! There is no multi-user checking going on.

Find a report that prints from a view whose performance is extremely slow, project a temporary table containing only the rows needed and drive the report from the temporary table. A report that takes 5-10 minutes to print might print in under a minute.

Flexibility

Because of the speed, you can do things you would never do with permanent tables. If you have systems that do an extraordinary amount of massaging of data placed in a temporary table. The work would take far longer (we are talking 10-100 times) to accomplish with a permanent table. And with permanent tables, deleting your scratch work takes a great deal of time and each record must have a user id in it to work correctly. With temp tables you just reconnect the database and all the temp tables are gone, just like that.

Temporary tables/views are also supported when STATICDB is set to ON.

No database growth

The data in the temporary tables is not part of File 2- the data file. The most important thing about temporary tables is that the actual data for a given user is written into a scratch file (.$$$). With an actual table that data is stored in the data file.

For example, if you are using an actual table and start with a 100MB data file and add 5MB of "temporary data" to a database, then drop the working table. Now, your data file is 105MB. Then, run the procedure again and you'll have 110MB. Running the procedure two more times and your data file is 120MB, and so on. You would have perform a PACK or RELOAD just to return back to the 100MB

Copyright © 1982-2024 R:BASE Technologies, Inc.

data file after processing temporary data within actual tables. On the other hand, if you use temporary tables your data file doesn't grow at all.

Views don't really make much difference since the only thing that would be stored in the database itself (and they still might be with temporary views) would be the structure. Everything else is generated when you actually use the view.

Independent

Temporary Tables/Views are specific to each session of R:BASE and that specific user.

For example, five different users or sessions, can create the same temporary table with the same name and not interfere with the others. Only the user that created the table can see/use it. So what it means is that 5 different uses can be using a running a report on the temporary table/view and all 5 users have different data in the table.

The "Sales Order" option in Running R:BASE Your Way! (Part 19), sample applications bundled with R:BASE, demonstrates the typical use of this feature.

Usability

You can treat a temporary table/view as a regular table/view. You can create Forms, Reports and Labels based upon the temporary table/view.

A powerful use of temporary tables is to PROJECT or CREATE a temporary table to collect (LOAD) data and allow easy editing prior to an INSERT. Since each session of R:BASE will project/create its own private temporary table (of the same name) this is an ideal solution, say for collecting some accounting data prior to allowing the user to post the transaction to the formal journal tables. As soon as the insert is done, a DISCONNECT/CONNECT will eliminate the temporary table and you are ready for next time.

Temporary tables are great when you are trying to take a huge vertical table with hundreds of thousands of rows and farm it out to some aggregate tables.

Sometimes when you need to insert row(s) into a table based on rows in the table, (the where clause cannot refer to the same table for the insert) You may project a temp table of the correct where conditions and insert where column in permanent table in (select column from temp table). You can do all kinds of variations of this one, such as using existing rows as a template which you house in a temp table, edit for the new values and re-insert back to the permanent table.

Disadvantages of Temporary Tables/View:

The advantage of temporary tables/views is also their disadvantage. They are not permanent. Any data you wish to be persistent must go in real tables/views.

1.39 Transaction Processing

Transaction processing allows you to process several commands as a group, or *transaction*. For each transaction, you can keep the changes made by the group of commands, or you can undo the changes, for instance if a command error occurs or a transaction cannot be completed.

Transaction processing works in both single and multi-user modes.

Transaction processing offers several advantages, the greatest of which is improved data integrity.

For example, if you are entering debits and credits, you can ensure that either all or none of the data is entered. Or suppose you are posting accounts at the end of the year, and deleting rows as they are posted to a master account. If the posting process is interrupted or a required table is not available, transaction processing lets you cancel the entire process and start over without having to reconstruct each original table.

Copyright © 1982-2024 R:BASE Technologies, Inc.

You can also use transaction processing to analyze data in a hypothetical situation. You can add data to a table, perform calculations, analyze the results, then decide whether to keep the new data. You can try different scenarios as many times as needed to find the right combination of data and keep only the data that produced the desired result.

Transaction processing has the disadvantage of decreasing system performance.

When you use it, R:BASE must keep track of extra table and database locks, maintain a Before Image file for each user, and perform internal consistency checks. Every transaction requires overhead, which slows the entire system.

Transaction Processing Topics:

1.39.1 Using Transaction Processing

To use transaction processing, enter the following command at the R> Prompt or in a command file before you open a database:

```
SET TRANSACT ON
```

While you are working at the R> Prompt, if you want each command to be made permanent immediately and visible to network users, turn AUTOCOMMIT on:

```
SET AUTOCOMMIT ON
```

If you want to be able to process a transaction without permanently affecting your data until you accept the changes, turn AUTOCOMMIT off. Then, to accept a transaction and make the changes permanent, enter the COMMIT command after you have entered the transaction. Or, to reverse the transaction and undo the changes, enter the ROLLBACK command. Whenever you disconnect from a database in the middle of a transaction (AUTOCOMMIT is set off), the transaction is saved as though you had entered a COMMIT command.

If you have started a transaction when you set AUTOCOMMIT on, R:BASE commits the transaction and turns AUTOCOMMIT on. You cannot open a cursor while AUTOCOMMIT is set on, and you cannot set AUTOCOMMIT on while a cursor is open. AUTOCOMMIT and MDI cannot be set to on at the same time. If MDI forms and browse screens are to be used, then autocommit mode cannot be running. If autocommit mode is required, then forms and browse screens cannot be modeless.

Because transaction processing also works in multi-user mode, you can specify the maximum number of users who can have the same database open concurrently with transaction processing on by specifying a value for MAXTRANS. For example, to specify 20 users:

```
SET MAXTRANS 20
```

In transaction processing, if you change the value of a SET command whose value is normally stored with the database, the new value is effective only until you close the database. The new SET value is not stored with the database.

With transaction processing on, execution time for commands is determined by several factors, particularly the number of users accessing the same database and the types of commands being executed. Some commands take longer to execute than others, and the same command can take longer from one user to the next.

Copyright © 1982-2024 R:BASE Technologies, Inc.

You cannot use the following commands with transaction processing ON because these commands create new database structures or start other modules:

ALTER TABLE	CODELOCK	RBAPP
FORMS	GATEWAY	PACK
RBDEFINE	RBLABELS	RELOAD
RENAME OWNER	REPORTS	UPGRADE

Transaction Processing Topics:

Locking Table Access and Resource Waiting
Row Locks and Transaction Processing
Automatic Table and Database Locks
Setting Exclusive Table Locks
Displaying Transaction Processing Locks
Resource Waiting in Transaction Processing
Generate a Transaction Journal
Recovering from Transaction Processing Errors

1.39.2 Locking Table Access and Resource Waiting

To ensure data integrity, R:BASE maintains a system of locks that control access to tables and databases.

When a command requires access to a resource, such as a table or database, R:BASE checks to see whether it is already in use. If it is locked, R:BASE checks whether the existing lock(s) prevent the new lock from being applied. If so, R:BASE puts the lock in a waiting queue. When the existing lock is removed, the next lock in the queue can access the table or database. While the command is waiting for a lock, you are prompted to continue waiting or to abort.

Transaction processing locks permit maximum concurrent access to system resources, yet maintain strict data integrity. The locks used by transaction processing replace those used in multi-user mode. Automatic locks are removed at the end of the transaction that required the lock.

Locks are not needed when transaction processing is used in single-user mode.

Transaction Processing Topics:

Using Transaction Processing
Row Locks and Transaction Processing
Automatic Table and Database Locks
Setting Exclusive Table Locks
Displaying Transaction Processing Locks
Resource Waiting in Transaction Processing
Generate a Transaction Journal
Recovering from Transaction Processing Errors

1.39.3 Row Locks and Transaction Processing

Rowlocks and fastlocks are incompatible with transaction processing; you cannot have either ROWLOCKS or FASTLOCKS set ON when transaction processing (TRANSACT) is set on. If you set transaction processing on when ROWLOCKS or FASTLOCKS are set on, R:BASE forces the ROWLOCKS and FASTLOCKS settings off. If you try to set ROWLOCKS or FASTLOCKS on when transaction processing is set on, R:BASE displays an error message.

Transaction Processing Topics:

Using Transaction Processing
Locking Table Access and Resource Waiting
Automatic Table and Database Locks
Setting Exclusive Table Locks
Displaying Transaction Processing Locks
Resource Waiting in Transaction Processing

Copyright © 1982-2024 R:BASE Technologies, Inc.

Generate a Transaction Journal
Recovering from Transaction Processing Errors

1.39.4 Automatic Table and Database Locks

The following list describes the automatic table and database locks used in transaction processing.

Transaction Processing Locks

- SELECT table lock

Commands	
BROWSE	COMPUTE
CROSSTAB	PRINT
QUERY	SELECT
TALLY	

These commands only need to view data from a table. A SELECT lock excludes all other locks except other SELECT locks.

- INSERT table lock

Commands	
ENTER	ENTER USING
INSERT	LOAD
OPEN CURSOR	

These commands may add a row to a table. An INSERT lock excludes all other locks.

- UPDATE table lock

Commands	
DELETE	EDIT
EDIT USING	UPDATE

These commands update information in a table. An UPDATE lock excludes all other locks.

- Full database lock

Commands	
AUTONUM	BACKUP
CREATE INDEX	CREATE TABLE
CREATE VIEW	DROP INDEX
DROP TABLE	DROP VIEW
GRANT	INTERSECT
JOIN	PROJECT
RENAME COLUMN	RENAME TABLE
RENAME TABLE	RENAME VIEW
RESTORE	REVOKE
RULES	SUBTRACT
UNION	UNLOAD

Copyright © 1982-2024 R:BASE Technologies, Inc.

A full database lock excludes locks on all tables and the database and disables the LIST command until the transaction is finished. If a command cannot lock all the tables in a database, you receive a message.

The commands are listed under the lock they obtain most frequently. The current use of each command, however, determines which lock it obtains. These commands can only upgrade their locks within a transaction.

Transaction Processing Topics:

Using Transaction Processing
Locking Table Access and Resource Waiting
Row Locks and Transaction Processing
Setting Exclusive Table Locks
Displaying Transaction Processing Locks
Resource Waiting in Transaction Processing
Generate a Transaction Journal
Recovering from Transaction Processing Errors

1.39.5 Setting Exclusive Table Locks

Transaction processing uses an exclusive lock. To activate an exclusive lock on a table, use the SET LOCK *tblname* ON command. This command remains in effect until a corresponding SET LOCK *tblname* OFF command is issued to remove the lock from that table. Only the user who sets the lock can remove it.

SET LOCK *tblname* OFF commands take effect when you end the current transaction with the COMMIT or ROLLBACK command. You should remove exclusive table locks as soon as the table is no longer needed, especially when working in multi-user mode. Exclusive locks are removed automatically when you exit a database, or when you run the recovery program following a processing interruption.

Transaction Processing Topics:

Using Transaction Processing
Locking Table Access and Resource Waiting
Row Locks and Transaction Processing
Automatic Table and Database Locks
Displaying Transaction Processing Locks
Resource Waiting in Transaction Processing
Generate a Transaction Journal
Recovering from Transaction Processing Errors

1.39.6 Displaying Transaction Processing Locks

You can use the LIST or LIST TABLES command to see which locks are in place; when transaction processing is on, the names of locked tables are listed with the LIST command in reverse video.

Types of locks displayed with the LIST TABLES command:

Lock	Description
Local SELECT lock	A command issued from this workstation holds a SELECT lock
Remote SELECT lock	A command issued from another workstation holds a SELECT lock
Local INSERT lock	A command issued from this workstation holds an INSERT lock
Remote INSERT lock	A command issued from another workstation holds an INSERT Lock
Local UPDATE lock	A command issued from this workstation holds an UPDATE lock
Remote UPDATE lock	A command issued from another workstation holds an UPDATE Lock
Local lock	SET LOCK issued from this workstation holds an exclusive lock
Remote lock	SET LOCK issued from another workstation holds an exclusive lock

Transaction Processing Topics:

Using Transaction Processing
Locking Table Access and Resource Waiting

Copyright © 1982-2024 R:BASE Technologies, Inc.

Row Locks and Transaction Processing
Automatic Table and Database Locks
Setting Exclusive Table Locks
Resource Waiting in Transaction Processing
Generate a Transaction Journal
Recovering from Transaction Processing Errors

1.39.7 Resource Waiting in Transaction Processing

You can determine how long R:BASE continues to retry the last command until it receives a resource lock. By default, R:BASE waits four seconds and tries to access the resource throughout the wait period. If the table or database is still unavailable when the wait period expires, a message is displayed. You can either retry the command, or roll back the transaction.

To change the wait period, specify a value in seconds for the WAIT setting; for example, enter:

```
SET WAIT 10
```

You can also specify how often R:BASE retries the command within the SET WAIT period. The maximum setting for the INTERVAL command (in tenths of a second) is nine; the default is five.

For example, to specify an interval of nine tenths of a second, enter:

```
SET INTERVAL 9
```

Transaction Processing Topics:

Using Transaction Processing
Locking Table Access and Resource Waiting
Row Locks and Transaction Processing
Automatic Table and Database Locks
Setting Exclusive Table Locks
Displaying Transaction Processing Locks
Generate a Transaction Journal
Recovering from Transaction Processing Errors

1.39.8 Generate a Transaction Journal

A transaction journal may be created when transaction processing is on, and the transaction journal setting, TJOURNAL, is also on.

SET TJOURNAL toggles journaling of commands that modify data when transaction processing is on.

The transaction journal log file is located in the same directory as the database. The file name consists of the database name followed by "_TJournal" with ".LOG" as the extension. For example, the RRBYW20 sample database would have a journal file with the name RRBYW20_TJournal.LOG.

Transaction Processing Topics:

Using Transaction Processing
Locking Table Access and Resource Waiting
Row Locks and Transaction Processing
Automatic Table and Database Locks
Setting Exclusive Table Locks
Displaying Transaction Processing Locks
Resource Waiting in Transaction Processing
Recovering from Transaction Processing Errors

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.39.9 Recovering from Transaction Processing Errors

You can use the RECOVER command to restabilize a database after a transaction is interrupted. The following conditions can interrupt a transaction:

- R:BASE discovers minor inconsistencies or unresolvable resource conflicts within a transaction. R:BASE automatically rolls back the transaction and displays an error message.
- Transaction processing has been interrupted unexpectedly. When you try to connect to the database, R:BASE detects either an existing Before Image file or major inconsistencies. An error message tells you to use the RECOVER command.

Other users connected to the same database might get the same message, but not until they disconnect from the database and try to reconnect to it. The transactions entered by other users still connected to the database are executed properly unless they try to access system resources (such as tables and views) that are directly affected by the transaction error.

When transaction processing is on, R:BASE creates Before Image files for the current database.

When you use the RECOVER command, it rolls back the interrupted transaction by using each Before Image file, clears all table and database locks, and resolves any other internal inconsistencies.

Before you run the recovery program, do the following:

- Backup your database and Before Image files.
- Make sure all users have exited from the affected database.
- Change to the directory that contains the affected database and Before Image files.

To use the recovery program:

1. At the R> Prompt, enter RECOVER. All the databases that need to be recovered are listed.
2. At the R> Prompt, enter RECOVER *dbname*, where *dbname* is the name of the database you want to recover.
3. R:BASE displays a message recommending that you backup the damaged database and Before Image files. If you want to continue, select Yes. If you want to stop the RECOVER command to backup the necessary files, select No.
4. RECOVER displays the names of any damaged Before Image files and asks whether to delete them in order to recover the rest of the database. To delete these files, select Yes. To skip the deletion, select No. However, you must delete damaged Before Image files before you can access the database.

   RECOVER displays a list of any Before Image files it expects but cannot find. You can leave the database unrecovered, or recover it without the missing files. You cannot reconstruct a Before Image file.

   A message is displayed when the recovery is complete. The recovery process removes all table and database locks that were active when processing was interrupted.

Transaction Processing Topics:

Using Transaction Processing
Locking Table Access and Resource Waiting
Row Locks and Transaction Processing
Automatic Table and Database Locks
Setting Exclusive Table Locks
Displaying Transaction Processing Locks
Resource Waiting in Transaction Processing
Generate a Transaction Journal

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.39.10 Example

Below is an example of a form being edited where committing a new customer to record is employed.

```
-- Transaction Mode Commit/Rollback
DISCONNECT
SET TRANSACT ON
SET AUTOCOMMIT OFF
CONNECT ConComp

CLEAR ALL VAR
SET VAR vTime TIME = .#TIME

INSERT INTO Customer LastUpdateTime VALUES (.vTime)

EDIT USING CustomerEdit WHERE LastUpdateTime=.vTime

DIALOG 'Is the Customer Information Being Saved to Record?' +
vYesNo vEndKey No CAPTION ' ' ICON QUESTION

IF vYesNo = 'YES' THEN
  COMMIT
ELSE
  ROLLBACK
ENDIF
RETURN
```

1.40 Using PAGEMODE

The following section provides examples of using PAGEMODE. For more information about the PAGEMODE command, see SET PAGEMODE.

There are common elements to all the PAGEMODE examples. LINES are set artificially low in each example to force a page break to occur. All the examples direct output to a file and the file is displayed to the screen when the report is complete. MESSAGES are set off to make sure they do not interfere with the report data.

Code Examples

1.40.1 Example: Testing for the End of a Page

Your program needs to be aware of the current setting for LINES so you can test for the end of a page. The current row count is compared to a constant value or to the LINES setting to check for the end of page. The CVAL function is used to place the current line setting in a variable. For example,

```
SET VARIABLE vLines = (CVAL('LINES'))
```

Then, an IF statement compares the *vLines* variable with the *vRowCount* variable. For example,

```
IF vRowCount >= .vLines THEN
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
  NEWPAGE
  SET VARIABLE vRowCount = 1
  SET VARIABLE vRow = 1
  SET VARIABLE vColumn = 1
ENDIF
```

The *vRowCount, vRow,* and *vColumn* variables must be reset for each new page. Leave a five line margin at the bottom of the page by comparing *vRowCount* to *vLines* minus five. For example,

```
IF vRowCount >= (.vLines - 5) THEN
  NEWPAGE
  SET VARIABLE vRowCount = 1
  SET VARIABLE vRow = 6
  SET VARIABLE vColumn = 5
ENDIF
```

Margins at the top of the page and on the left side are determined by the initial settings for the *vRow* and *Vcolumn* variables.

You can calculate the number of rows to print for the next data item, and compare that to see if all the data will fit on a page. This calculation is used to make sure break header, detail, and break footer information all appear on the same page.

```
SELECT COUNT(*) INTO vNumToPrint +
  FROM transmaster WHERE custid = .vCustid
IF (.vRowCount + .vNumToPrint) >= .vLines THEN
  NEWPAGE
  SET VARIABLE vRowCount = 1
  SET VARIABLE vRow = 6
  SET VARIABLE vColumn = 5
ENDIF
```

There are many different ways to keep track of the number of rows that have been printed on the page. The important thing to remember is that you need to keep track of it and issue the form feed yourself; it does not happen automatically.

1.40.2 Example: Phone List with Breakpoints

This example prints a phone list report using breakpoints. Each break is printed on a separate page. The data is not printed in columns to make it easier to see how breakpoints are set up. The employee phone list data is printed by state.

```
*(PHONE2.RMD)
*(example 2 - a report with break points.
  An employee phone list is printed by state,
  each state on a separate page.)
DEL phone.out
CON hifi
CLS
SET MESSAGE OFF
SET PAGEMODE OFF
SET LINES 15
SET VAR vState TEXT
DECLARE c1 CURSOR FOR SELECT DISTINCT state +
  FROM salespeople
DECLARE c2 CURSOR FOR +
  SELECT (LastName + ',' & FirstName), Phone, Lastname +
  FROM salespeople WHERE state = .vState +
  ORDER BY lastname
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
SET VAR vCol INT = 3, +
        vRow INT = 2, +
        vCount INT = 0, +
        vPage INT = 1, +
        vPageLimit = (CVAL('LINES'))
SET PAGEMODE ON
OPEN c1
FETCH c1 INTO vState ii1
WHILE SQLCODE <> 100 THEN
OUTPUT phone.out APPEND
WRITE 'List of employees in state of:', .vState, +
  '                      Page:', .vPage AT .vRow, .vCol
SET VAR vRow = (.vRow + 1)
WRITE '---------------------------------' +
    AT .vRow, .vCol
OPEN c2
  FETCH c2 INTO vName i1 ,vPhone i2, vLastname i3
  WHILE SQLCODE <> 100 THEN
    SET VAR vRow = (.vRow + 1)
    SET VAR vCount = (.vCount + 1)
    IF vRow >= .vPageLimit THEN
      NEWPAGE
      SET VAR vRow = 2, vPage = (.vPage + 1)
      WRITE 'List of employees in state of:', .VState, +
  '                      Page:', .VPage AT .vRow, .VCol
      SET VAR vRow = (.vRow + 1)
      WRITE '---------------------------------' +
        AT .vRow, .vCol
    ENDIF
    WRITE .vName=21 .vPhone AT .vRow .vCol
    FETCH c2 INTO vName i1, vPhone i2, vLastname I3
  ENDWH
  CLOSE c2
  SET VAR vRow = (.vRow + 2)
  WRITE 'Number of employees for', .vState, 'is', +
    .vCount AT .vRow, 10
  OUTPUT SCREEN
  SET VAR vRow = 3, vPage = (.vPage + 1), vCount = 0
  FETCH c1 INTO vState ii1
ENDWH
OUTPUT screen
DROP CURSOR c1
DROP CURSOR c2
SET PAGEMODE OFF
SET LINES 20
TYPE phone.out
```

1.40.3 Example: Multiple Breaks

```
-- Initialize variables

SET VAR vCustState TEXT, vCustid TEXT

-- Set up the breakpoints with cursors.
-- Cursor c1 is break 1, cursor c2 is break 2,
-- cursor c3 retrieves the detail data.
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
DECLARE c1 CURSOR FOR +
  SELECT DISTINCT coststate FROM customer
DECLARE c2 CURSOR FOR +
  SELECT DISTINCT custid FROM transmaster +
  WHERE custid IN +
       (SELECT custid FROM customer +
        WHERE coststate = .vCustState)
DECLARE c3 CURSOR FOR +
  SELECT transid, empid, invoicetotal, transdate +
  FROM transmaster WHERE custid = .vCustid
OPEN c1
FETCH c1 INTO vCustState Ind1
WHILE SQLCODE <> 100 THEN

  -- This is the break header 1 position,
  -- the first state value is retrieved.
  -- Cursor c2 then fetches all customer rows
  -- for that particular state

  OPEN c2 RESET
  FETCH c2 INTO vCustid Ind2
  WHILE SQLCODE <> 100 THEN

    -- This is the break header 2 position,
    -- the first customer row. Cursor c3 then
    -- fetches all transaction rows for
    -- that customer

    OPEN c3 RESET
    FETCH c3 INTO vTransid vind1, vEmpid vind2, +
      vInvoiceTotal vind3, vTransDate vind4
    WHILE SQLCODE <> 100 THEN

      -- The detail rows are processed here

      FETCH c3 INTO vTransid vind1, vEmpid vind2, +
        vInvoiceTotal vind3, vTransDate vind4
    ENDWHILE

    -- Break footer 2 position is here, +
    -- right before the next
    -- customer row is fetched.

    FETCH c2 INTO vCustid Ind2
  ENDWHILE

  -- Break footer 1 position is here, after
  -- all the customer data has been fetched,
  -- before the next state value is fetched.

  FETCH c1 INTO .vCustid
ENDWHILE
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.40.4 Example: Employee Telephone List

This example prints an employee telephone list in columnar format. The page header data is actually written at the end of the page so it includes data from the first row printed on the page, and data from the last row printed on the page.

```
*(PHONE1.RMD)
*(example 1 - a columnar report with a page header
  written at the end of the page)
CON hifi
CLS
SET MESSAGE OFF
SET PAGEMODE OFF
SET LINES 10
DECLARE c1 CURSOR FOR +
  SELECT (LastName + ',' & FirstName), Phone, Lastname +
  FROM salespeople ORDER BY lastname
OPEN c1
FETCH c1 INTO vName i1 ,vPhone i2, vLastname I3
SET VAR vCol INT = 1, vRow INT = 3, vRowLimit = 9
SET VAR vFirst TEXT = .vLastname, +
        vLast TEXT = .vLastname,+
        vPage INT = 1
SET PAGEMODE ON
OUTPUT phone.out
WHILE SQLCODE <> 100 THEN
   SET VAR vRow = (.vRow + 1)
   IF vRow >= .vRowLimit THEN
     IF vCol = 40 THEN
       WRITE .vFirst '-' .vLast AT 2 1
       WRITE 'Page' .vPage AT 2 68
       WRITE '----------------------------------' AT 3 1
       WRITE '----------------------------------' AT 3 40
       NEWPAGE
       SET VAR vFirst = .vLastname, vLast = .vLastname
       SET VAR vPage = (.vPage + 1), vCol = 1
     ELSE
       SET VAR vCol = 40
     ENDIF
     SET VAR vRow = 4
   ENDIF
   WRITE .vName=21 .vPhone AT .vRow .vCol
   SET VAR vLast = .vLastname
   FETCH c1 INTO vName i1, vPhone i2, vLastname I3
ENDWH
WRITE .vFirst '-' .vLast at 2 1
WRITE 'Page' .vPage at 2 68
WRITE '----------------------------------' at 3 1
WRITE '----------------------------------' at 3 40
OUTPUT SCREEN
CLOSE c1
DROP CURSOR c1
SET PAGEMODE OFF
SET LINES 20
TYPE phone.out
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.40.5 Example: Breakpoint Report

This is an example of a breakpoint report with multiple breaks printed per page. The break heading is repeated on the next page if the data for the break does not fit on the page. The report prints a page header as well as a break header and break footer.

```
*(PHONE3.RMD)
*(example 3 - this example includes a page header
  along with the break header. Multiple breaks are
  printed per page. The break header is repeated on
  subsequent pages if the break detail spans a page)
DEL phone.out
CON hifi
CLS
SET MESSAGE OFF
SET PAGEMODE OFF
SET LINES 25
SET VAR vState TEXT
DECLARE c1 CURSOR FOR SELECT DISTINCT state +
  FROM salespeople
DECLARE c2 CURSOR FOR +
  SELECT (LastName + ',' & FirstName), Phone, Lastname +
  FROM salespeople WHERE state = .vState +
  ORDER BY lastname
SET VAR vCol INT = 3, +
        vRow INT = 2, +
        vPage INT = 1, +
        vPageLimit = (CVAL('lines')), +
        vCount INT = 0
SET PAGEMODE ON
OUTPUT phone.out
WRITE 'Employee Phone List by State' AT .vRow, 25
SET VAR vRow = (.vRow + 1)
WRITE .#date AT .vRow, 32
WRITE 'Page:', .VPage AT .vRow, 60
SET VAR vRow = (.vRow + 2)
OPEN c1
FETCH c1 INTO vState ii1
WHILE SQLCODE <> 100 THEN
  WRITE 'List of employees in state of:', .VState +
    AT .vRow, .VCol
  SET VAR vRow = (.vRow + 1)
  WRITE '---------------------------------' +
    AT .vRow, .vCol
  OPEN c2
  FETCH c2 INTO vName i1 ,vPhone i2, vLastname i3
  WHILE SQLCODE <> 100 THEN
    SET VAR vRow = (.vRow + 1)
    SET VAR vCount = (.vCount + 1)
    IF vRow >= .vPageLimit THEN
      NEWPAGE
      SET VAR vRow = 2, vPage = (.vPage + 1)
      WRITE 'Employee Phone List by State' +
        AT .vRow, 25
      SET VAR vRow = (.vRow + 1)
      WRITE .#date AT .vRow, 32
      WRITE 'Page:', .vPageAT .vRow, 60
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
      SET VAR vRow = (.vRow + 2)
      WRITE 'List of employees in state of:', .VState +
        AT .vRow, .VCol
      SET VAR vRow = (.vRow + 1)
      WRITE '---------------------------------' +
        AT .vRow, .vCol
      SET VAR vRow = (.vRow + 1)
    ENDIF
    WRITE .vName=21 .vPhone AT .vRow .vCol
    FETCH c2 INTO vName i1, vPhone i2, vLastname i3
  ENDWH
  CLOSE c2
  SET VAR vRow = (.vRow + 2)
  WRITE 'Number of employees for', .vState, 'is', +
    .vCount AT .vRow, 10
  SET VAR vRow = (.vRow + 3), vCount = 0
  FETCH c1 INTO vState ii1
ENDWH
OUTPUT screen
DROP CURSOR c1
DROP CURSOR c2
SET PAGEMODE OFF
SET LINES 20
TYPE phone.out
```

1.40.6 Example: Checking Column Width

The command lines below are used to display data in three columns evenly spaced across the page with a four space margin on the left side of the page. You determine in your program whether the columns go across the page and then down, or first down the page and then across.

You should set up the column positions at the beginning of your program rather than automatically incrementing the column display variable by a certain amount and checking against the WIDTH setting.

```
SET VARIABLE vColumn1 = 5
SET VARIABLE vColumn2 = 30
SET VARIABLE vColumn3 = 55
 ...
WRITE .vName AT .vRow, .vColumn1
WRITE .vName AT .vRow, .vColumn2
WRITE .vName AT .vRow, .vColumn3
```

The number of columns can be dynamically calculated based on the maximum length of values to display for a group of data. Use the SLEN and MAX functions to find the maximum data length for the group of data.

```
-- compute the maximum text length

SELECT MAX(SLEN(title)) INTO vLen +
  FROM books WHERE lstnm = .vLName

-- set the number of columns based
-- on the maximum text length

IF vLen <= 25 THEN
  SET VAR vNumCols = 3
  SET VAR vColumn1 = 5
  SET VAR vColumn2 = 30
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
  SET VAR vColumn3 = 55
ELSE
  IF vLen <= 50 THEN
    SET VAR vNumCols = 2
    SET VAR vColumn1 = 5
    SET VAR vColumn2 = 40
  ELSE
    SET VAR vNumCols = 1
    SET VAR vColumn1 = 5
  ENDIF
ENDIF
```

Set SELMARGIN to specify the beginning column for SELECT command output. SELECT and SELMARGIN can be used with WHERE CURRENT OF CURSOR to display a row of data starting at any column position.

1.40.7 Example: Break Header/Footer

```
SET VAR vCustid TEXT
DECLARE c1 CURSOR FOR SELECT DISTINCT custid +
  FROM transmaster
DECLARE c2 CURSOR FOR SELECT transid, empid, +
  invoicetotal, transdate FROM transmaster +
  WHERE custid = .vCustid
OPEN c1
FETCH c1 INTO vCustid Ind1
WHILE SQLCODE <> 100 THEN

 -- This is the break header position,
 -- the first customer ID is retrieved.
 -- Cursor c2 fetches all transaction rows for
 -- that customer.

  OPEN c2 RESET
  FETCH c2 INTO vTransid vind1, vEmpid vind2, +
    vInvoiceTotal vind3, vTransDate vind4
  WHILE SQLCODE <> 100 THEN
     -- The detail rows are processed here.
    FETCH c2 INTO vTransid vind1, vEmpid vind2,
      vInvoiceTotal vind3, vTransDate vind4
  ENDWHILE

 -- The break footer position is here,
 -- right before the next customer row is fetched.

  FETCH c1 INTO .vCustid
ENDWHILE
```

1.41 Using the Advanced Rich Text Editor

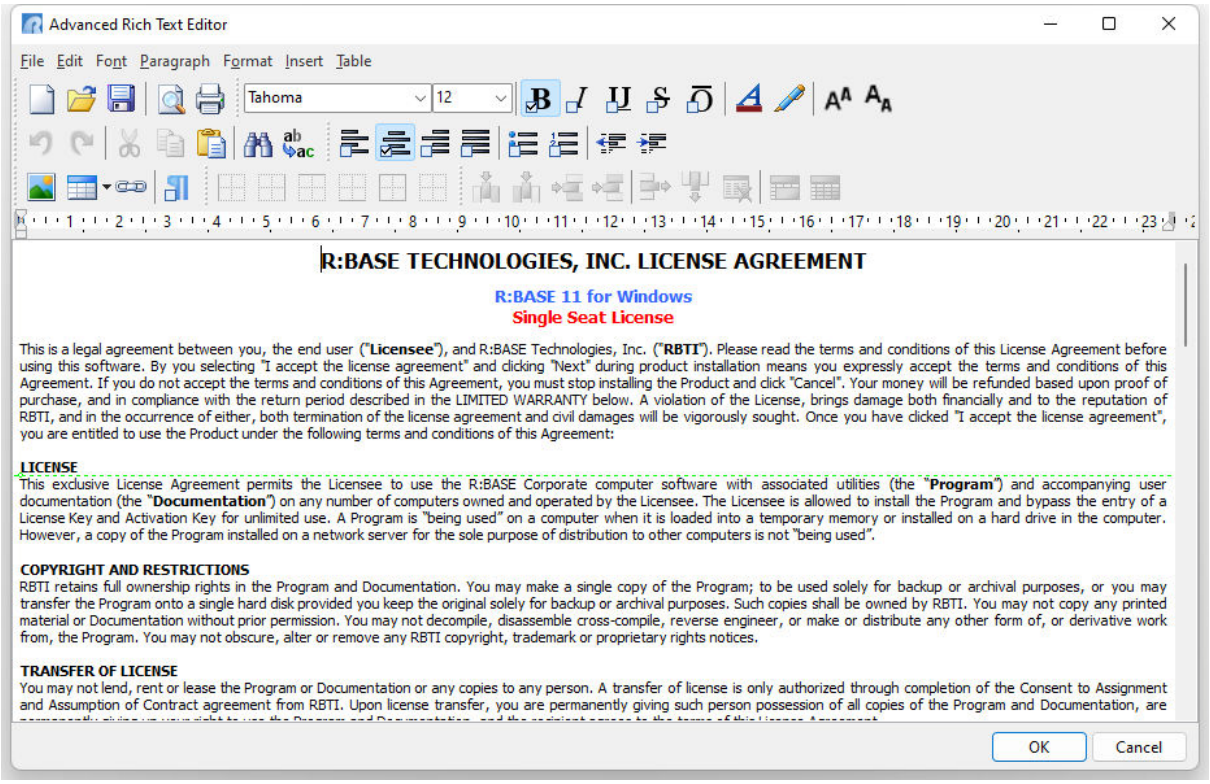
The Advanced Rich Text Editor is the new graphic user interface for working with Rich Text data in R:BASE. The Advanced Rich Text Editor allows for the rich formatting of text, images, tables,hyperlinks, lists, with ease-of-use manipulation with drag-and-drop and sizing support for content. The editor's toolbar is extensive allowing users to achieve a high degree of customization.

The editor can be accessed from within R:BASE forms using the Advanced DB Rich Text Control. In reports, there are three specific "Advanced Rich Text" controls that allow the placement of rich text with the advanced formatting the control supplies. The Advanced Rich Text Editor is also built into the R:BASE BLOB Editor when creating or editing rich text data.

Copyright © 1982-2024 R:BASE Technologies, Inc.

When added to R:BASE reports, the built-in "Mail Merge" option can be used to insert R:BASE columns and variables with text to create table driven output.

The Advanced Rich Text Editor can also be used to import existing files into R:BASE or even create rich text files for external file use.



Other features include:

Text Formatting
- Paragraphs , Borders, Backgrounds, Indents, Spacing, etc.
- Select the font face and font size and font color for selected text
- Standard formatting (Bold, Italic, Underline, Strike out, Overline)
- Left, Center, Right and Full Alignment of selected text
- Remove formatting from selected text
- Indent and outdent selected text

Table Formatting
- Insert tables
- Insert table row before current row
- Insert table row after current row
- Insert table column before current column
- Insert table column after current column
- Delete current column
- Delete current row
- Edit table properties

Hyperlinks
- Create hyperlinks
- Remove hyperlinks
- Edit Hyperlinks

Images
- Insert images

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Change layout properties such as alignment and spacing

Lists
- Insert numbered lists
- Insert bulleted lists

Other Functionality
- Insert R:BASE columns
- Insert R:BASE variables
- Insert horizontal lines
- Insert symbols from character map
- Undo/Redo
- Cut, copy and paste
- Select all
- Print
- Print Preview
- Page Setup
- Find and Replace
- Toggle the display of not-printing characters

1.41.1 Menu Bar

File

- *New* - allows you to create a new file
- *Open* - opens an existing file
- *Save* - saves the current file
- *Save As...* - saves the current file as a new file
- *Export* - exports the current file to an RTF or text file
- *Print Setup...* - opens the Print Setup dialog for changing page specific settings
- *Print Preview* - opens the Print Preview dialog
- *Print* - prints the current file to the default printer

Edit

- *Undo* - undoes the last change made
- *Redo* - reverses the effect of the last undo
- *Cut* - cuts the currently selected object(s)
- *Copy* - copies the currently selected object(s)
- *Paste* - pastes the currently cut or copied object(s)
- *Paste Special* - pastes text as Richview, rich text, text or Unicode format
- *Find...* - opens Find Text dialog box to locate text
- *Find Next* - locates the next text where matches search
- *Replace...* - opens Replace Text dialog box to locate and replace text
- *Character Case* - alters the case of the selected text
- *Insert Page Break* - adds a page break
- *Remove Page Break* - removes the page where the cursor is located
- *Select All* - selects all of the items

Font

- *Font* - alters a text font type, size, style, effect, character spacing, horizontal spacing, and offset
- *Style* - alters the select text style
- *Size* - alters the selected text size
- *Text Color* - alters the text color
- *Text Background Color* - alters the text background color
- *Superscript* - alters the text to appear as superscipt type
- *Subscript* - alters the text to appear as subscipt type

Paragraph

- *Paragraph* - specifies paragraph indents, spacing, tabs, and text flow
- *Paragraph Borders and Background* - specifies paragraph border options and color settings
- *Align Left* - aligns text to the left

Copyright © 1982-2024 R:BASE Technologies, Inc.

- *Align Center* - aligns text to the center
- *Align Right* - aligns text to the right
- *Justify* - justifies text to fit within the window
- *Bullets and Numbering* - allows customized options for bulleted and numbered lists
- *Bullets* - adds list bullets to the selected text
- *Numbering* - adds list numbering to the selected text
- *Word Wrap* - specifies if text is wrapped within the window
- *Decrease Indent* - decreases the indent for text
- *Increase Indent* - increases the indent for text
- *Single Line Spacing* - enables single line spacing
- *1.5 Line Spacing* - enables 1.5 line spacing
- *Double Line Spacing* - enables double line spacing

Format

- *Background* - adds an image to the background
- *Background Color* - specifies the entire background color
- *Fill Color* - specifies the fill color
- *Object Properties* - used to edit Table Properties

Insert

- *File* - inserts a rich text or text file
- *Picture* - inserts an image
- *Horizontal Line* - inserts a horizontal line
- *Hypertext Link* - adds a hypertext link
- *Symbol* - inserts a character symbol
- *DB Label* - inserts a database label for a column
- *Variable Label* - inserts a variable label for a defined variable

Table

- *Insert Table* - inserts a table
- *Insert Column Left* - inserts a column to the left of the cursor within the current table
- *Insert Column Right* - inserts a row to the right of the cursor within the current table
- *Insert Row Above* - inserts a row above the cursor within the current table
- *Insert Row Below* - inserts a row below the cursor within the current table
- *Delete Rows* - deletes the selected rows
- *Delete Columns* - deletes the selected columns
- *Delete Table* - deletes the current table
- *Select* - allows a table, the columns, rows, or all to be selected
- *Align* - allows alignment options
- *Cell Borders* - toggles the display of the table cell borders
- *Split Cells* - splits a current cell
- *Merge Cells* - merges the selected cells
- *Show Grid Lines* - toggles the display of the table grid lines
- *Table Properties* - displays the Table Properties

1.41.2 Toolbars

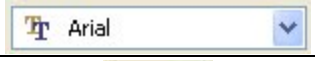
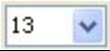
Button	Description
	New
	Open
	Save
	Print Preview
	Print

Button	Description
	Insert Column Left
	Insert Column Right
	Insert Row Above
	Insert Row Below
	Delete Column

Copyright © 1982-2024 R:BASE Technologies, Inc.

	Find
	Replace
	Cut
	Copy
	Paste
	Undo
	Redo
	Picture
	Insert Table
	Hypertext Link
	Nonprinting Characters

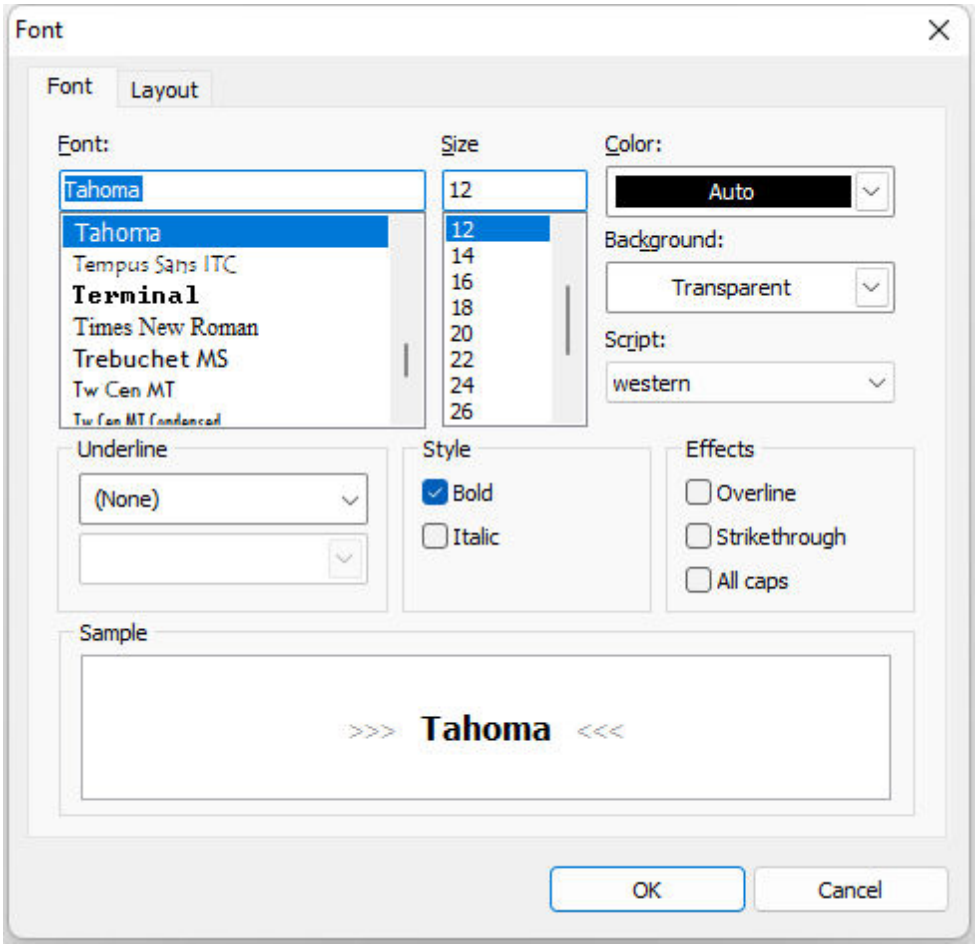
	Delete Table
	Left Border
	Right Border
	Top Border
	Bottom Border
	All Borders
	No Borders

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

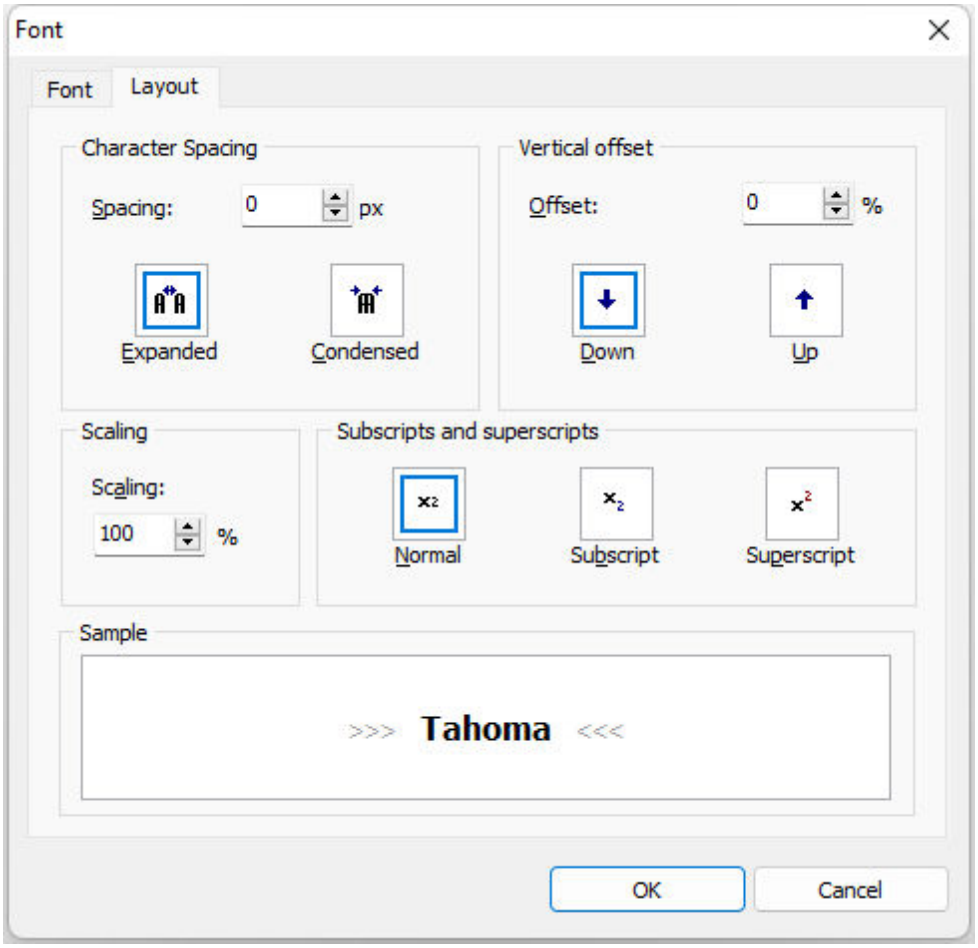
1.41.3 Font

The Font dialog allows the editing of font type, size, style, and effects.



Copyright © 1982-2024 R:BASE Technologies, Inc.

Layout options include character spacing, horizontal scaling, and offsetting up and down.

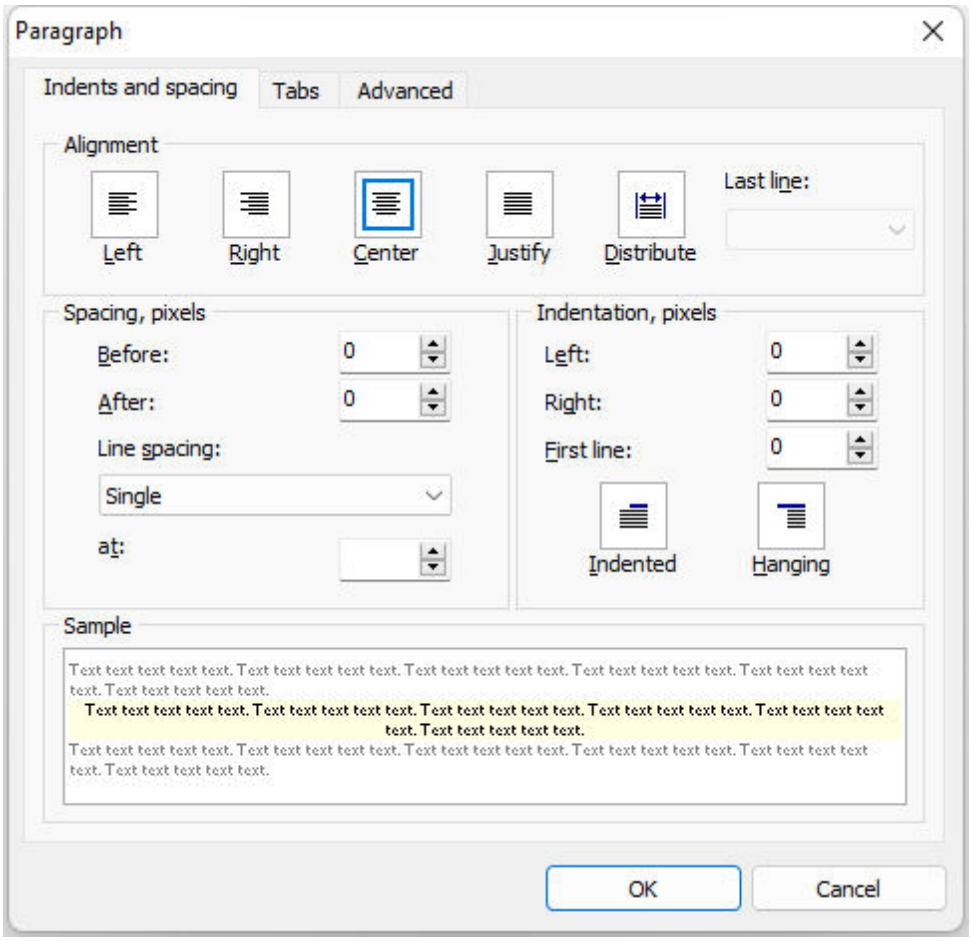


1.41.4 Paragraph

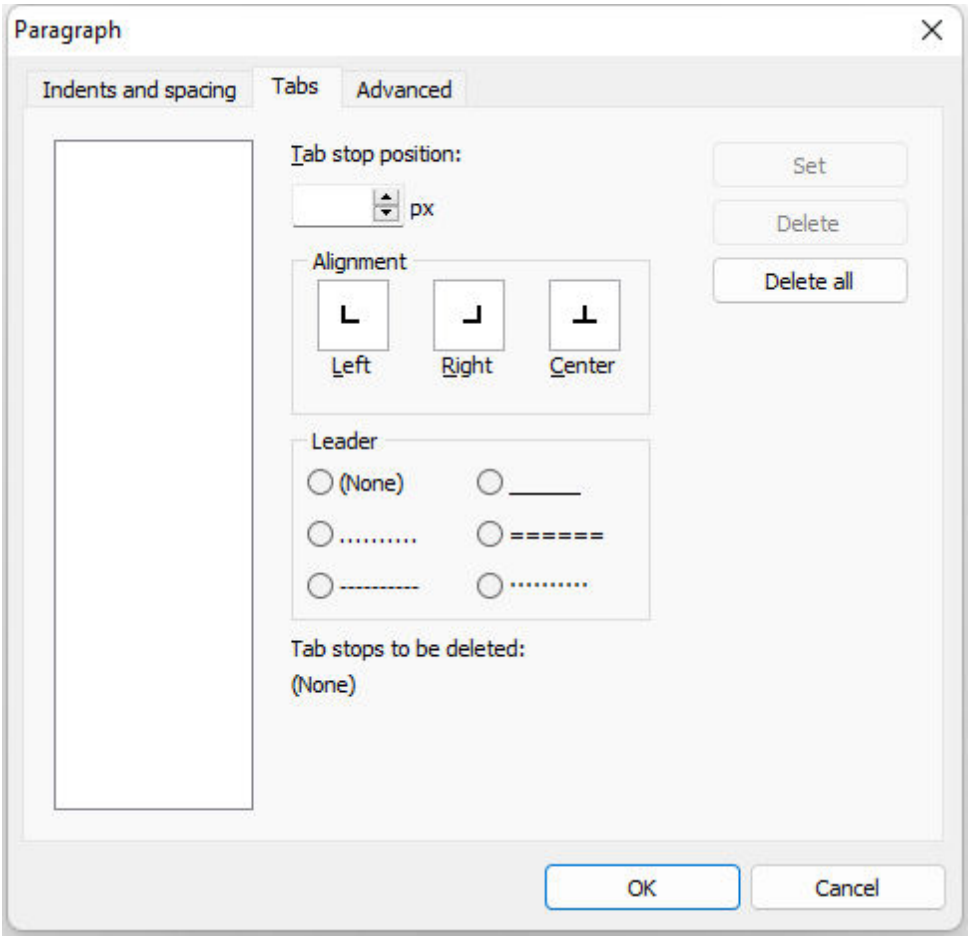
The Paragraph settings provide several display options for how sentence text is displayed.

Options for an entire paragraph include text alignment, spacing before and after the text, and indentation from the left and rights sides as well indenting for the first sentence of the paragraph. Line spacing can also be set.

Copyright © 1982-2024 R:BASE Technologies, Inc.

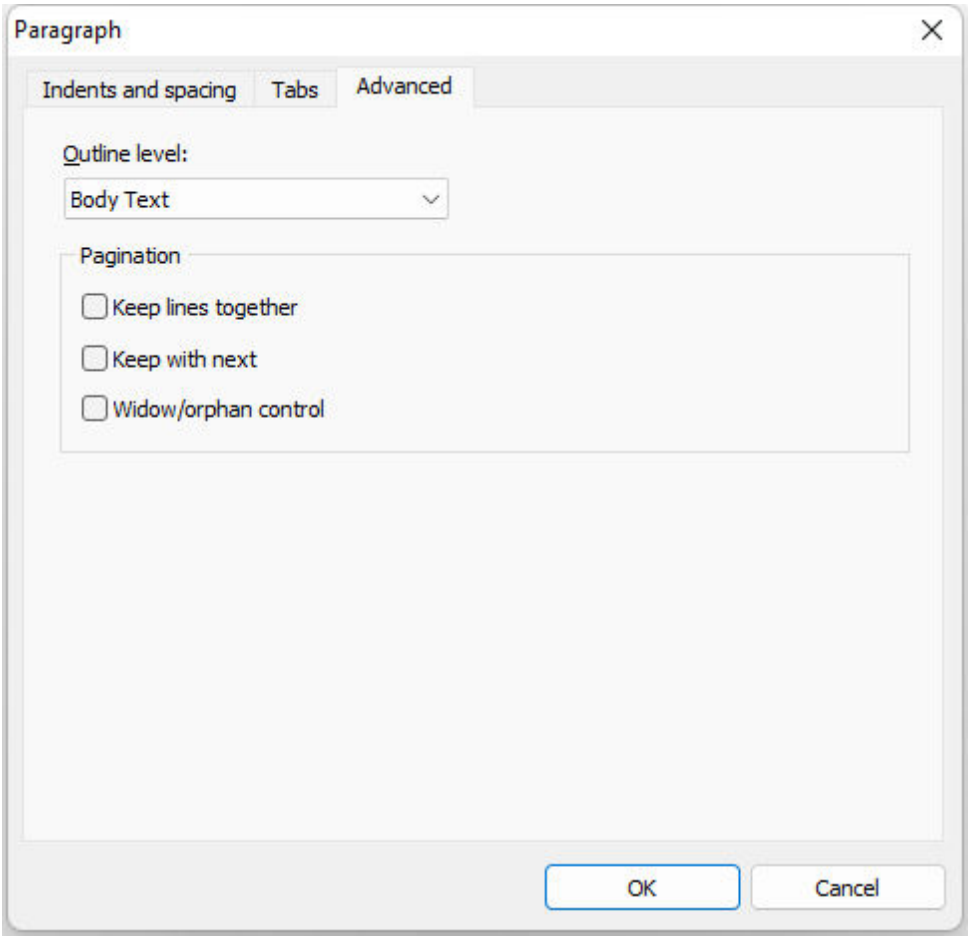


Copyright © 1982-2024 R:BASE Technologies, Inc.



The Tabs are special symbols that can be used for lining up text or numbers in tables, or for aligning text to a specific location for the rich text.

Copyright © 1982-2024 R:BASE Technologies, Inc.

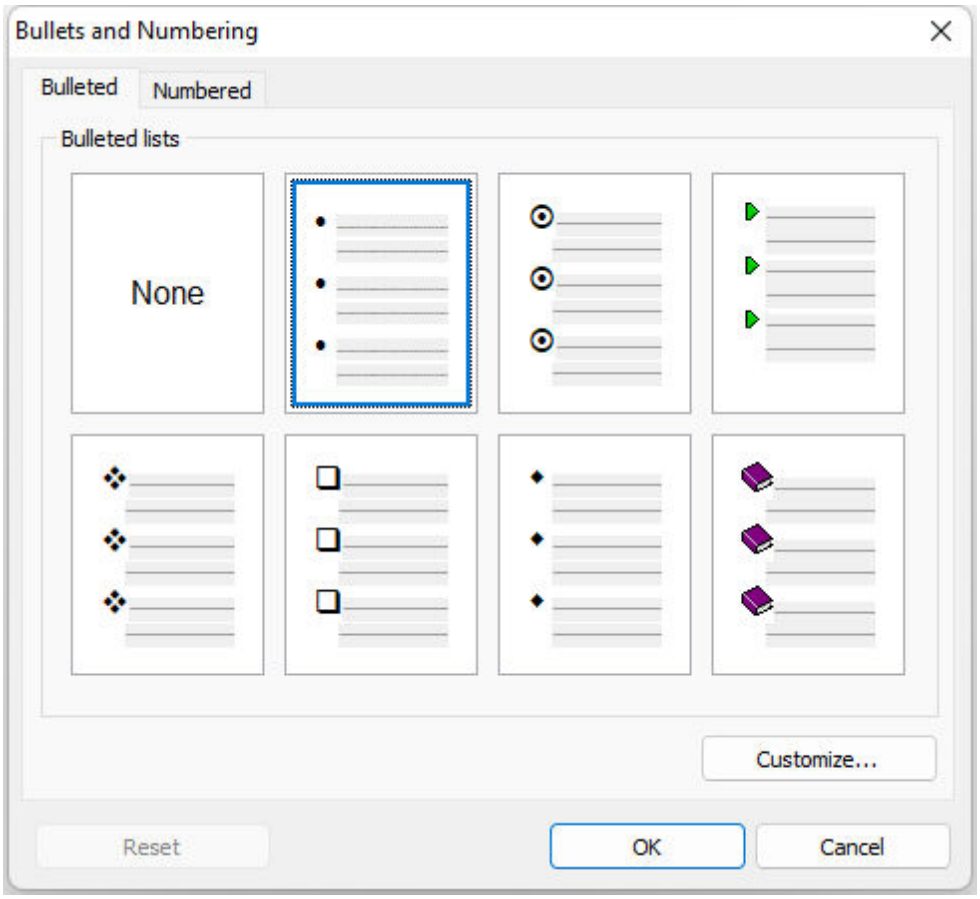


Advanced settings are available to control paragraph pagination to keep it on one page and to keep it on the same page with next paragraph.

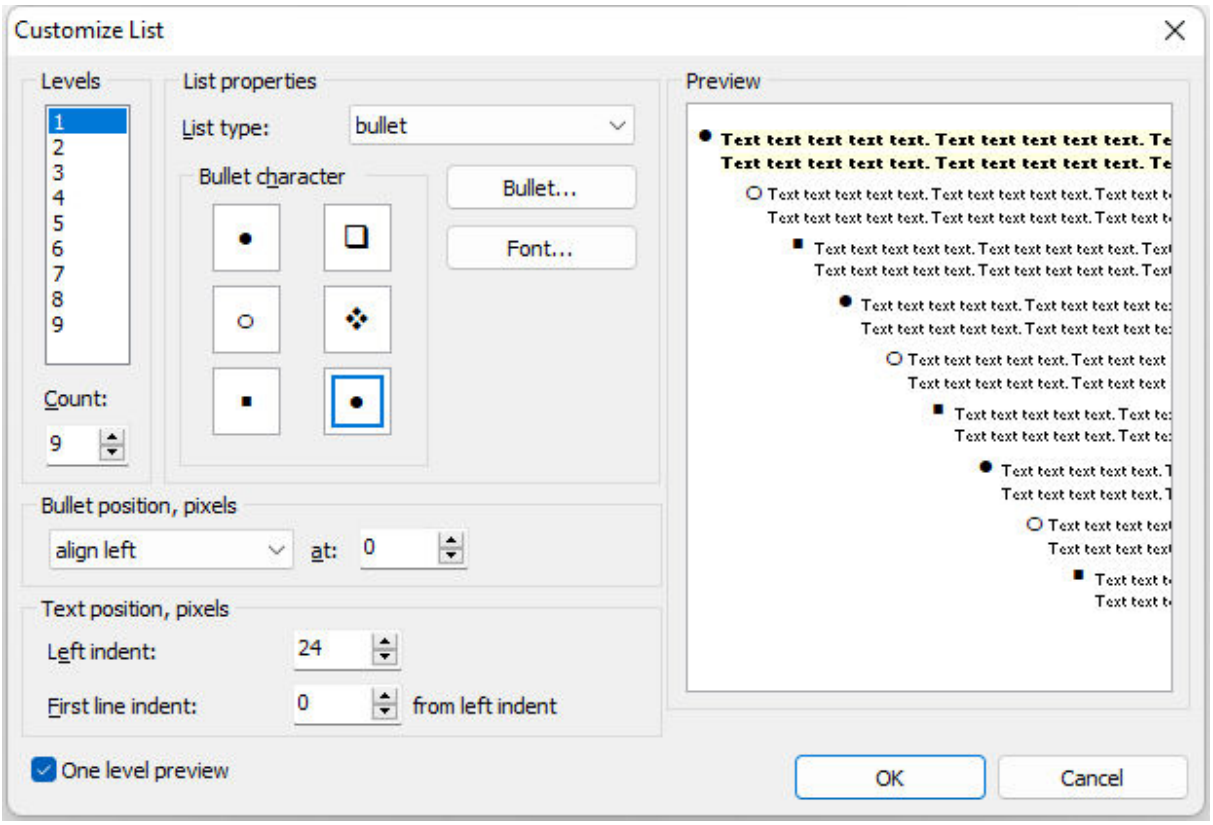
1.41.5 Bullets and Numbering

The Bullets and Numbering settings provide several display options for how lists are displayed. For bulleted lists, the bullet itself can be changed to display any of the below images.

Copyright © 1982-2024 R:BASE Technologies, Inc.

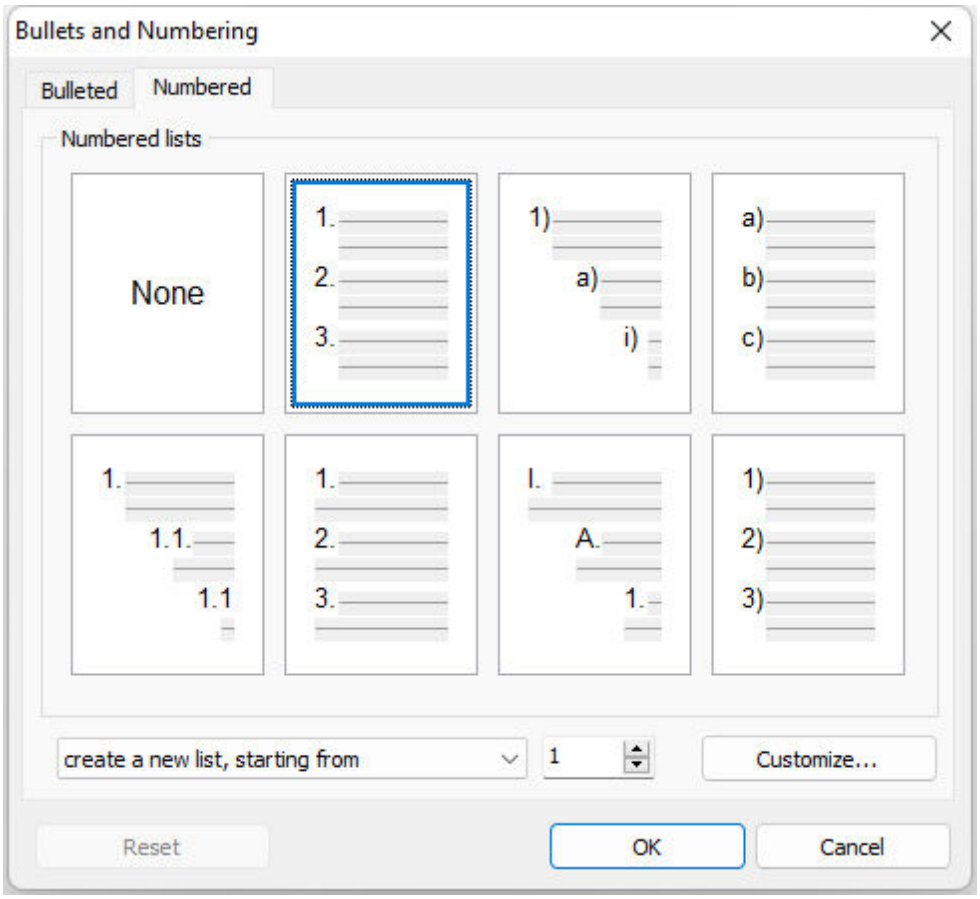


Copyright © 1982-2024 R:BASE Technologies, Inc.



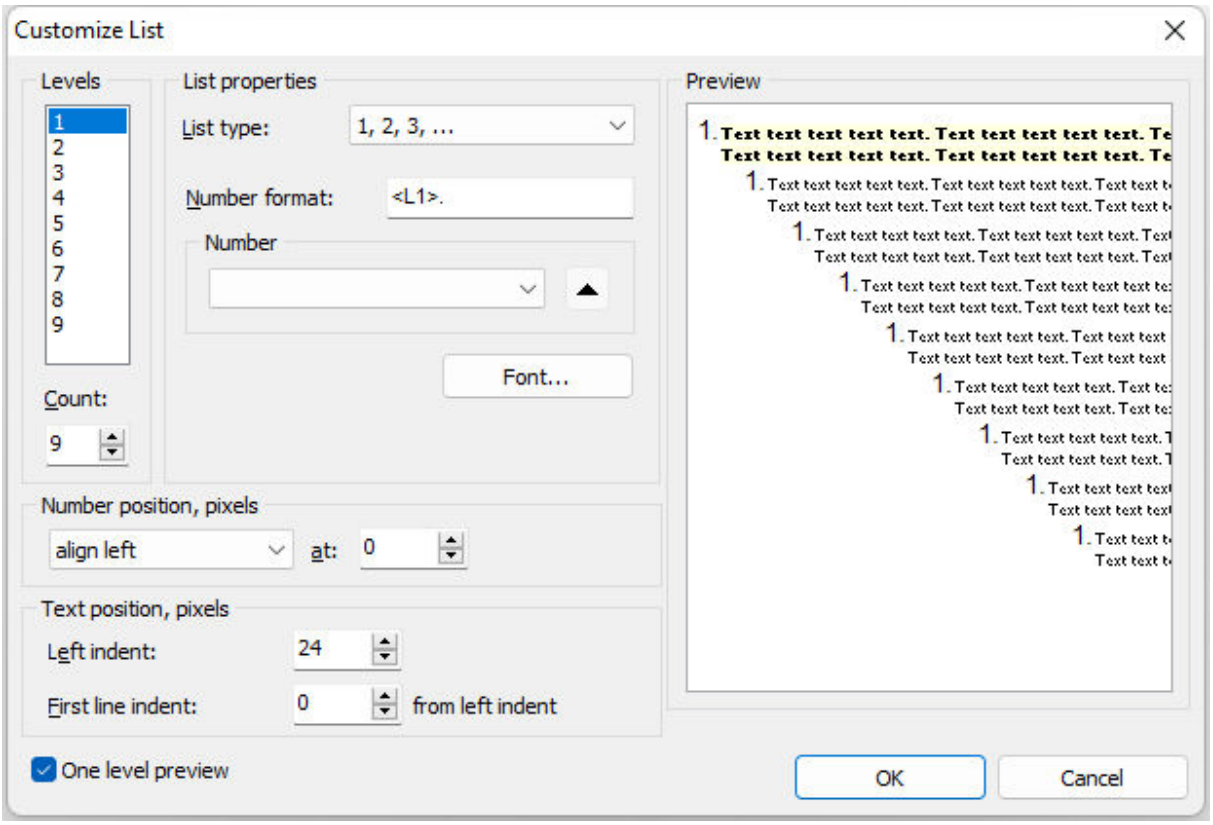
The "Customize..." button offers many additional display options to alter the levels, count, character, position, font, and indentation.

Copyright © 1982-2024 R:BASE Technologies, Inc.



For numbered lists, the numbers can be changed to display any of the below formats.

Copyright © 1982-2024 R:BASE Technologies, Inc.



The "Customize..." button offers many additional display options to alter the levels, count, format, position, font, and indentation.

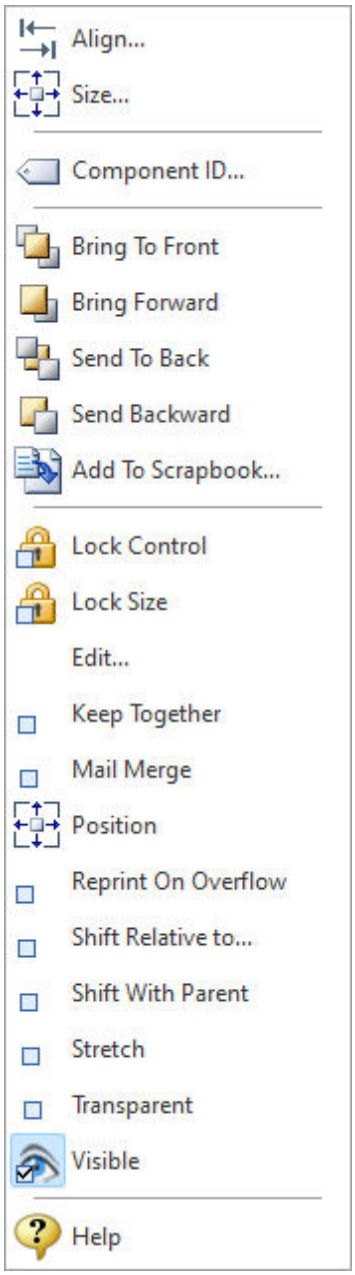
1.41.6 Context Menu

After placing the Advanced Rich Text control, right clicking on it will display a context menu with several options.

- **Align** - Displays the Alignment dialog for adjusting the horizontal and vertical alignment of the selected objects.
- **Component ID** - Unique identifier used when passing PROPERTY/GETPROPERTY parameters in statements calling the report
- **Bring to Front** - Places the currently selected object on the foreground of layered objects
- **Send to Back** - Places the currently selected object in the background of layered objects
- **Edit** - Opens the Advanced Rich Text Editor dialog for editing, loading, and saving text for the object
- **Mail Merge** - When checked, this property will cause the Rich Text component to scan the text, searching for occurrences of valid Data Pipeline field names surrounded by less than/greater than symbols (e.g. <custno>). If field names are found, they will be replaced with the field value. Only column names from the table which the report is based on are supported.

Copyright © 1982-2024 R:BASE Technologies, Inc.

- **Position** - Opens the Position dialog window for specifying hard-coded coordinates
- **Reprint On OverFlow** - Reprint On OverFlow applies to situations where a Stretchable component is stretching across pages. If the text overflows onto a new page, any components that are on the same band will reprint on the new page (if Reprint On OverFlow is checked).
- **Shift Relative To** - When multiple stretching components occupy the same band, use this property to specify how each component should be positioned relative to other stretching components. Shift Relative To can only be used for components that are dynamically sized. If you have a static component, such as a label, which needs to shift relative to a given stretching component, then place the component in a region and set the Region's Shift Relative To to the stretching component. **See Notes.**
- **Shift With Parent** - Shift With Parent applies to situations where a Stretchable component is stretching. If Shift With Parent is checked, the report component will move based on the amount of stretching the text requires. When you want components to appear at the bottom of a Rich Text object, this is a useful feature. If the object stretches to a new page, the components will print on the next page.
- **Stretch** - The Stretch property determines whether the Height of the object automatically stretches to allow the entire contents of the Rich Text to be printed.
- **Visible** - The Visible property determines whether a report component will be printed



Notes
When stretching components are placed within a region, the Shift Relative To property is disabled. You may notice that all of the stretching components in the band do not always appear in the Shift Relative To dialog when you are assigning this property. This is due to validation logic which ensures that components which are shifting relative to one another are not involved in a circular reference. In other words, if

Copyright © 1982-2024 R:BASE Technologies, Inc.

Memo1 is shifting relative to Memo2, then Memo2 cannot shift relative to Memo1, as this would create a situation which the report engine could not resolve.

1.41.7 Using Rich Text Controls to Mail Merge

The Advanced Rich Text Control in the R:BASE Report Designer includes the option to create Mail Merge reports.

This chapter will show you how to do the following:

- Generate a form letter with address information
- Edit a standard Rich Text document
- Use the Mail Merge feature of the Rich Text controls

Create a Report

1. Connect to a database that has a table which contains addresses
2. Create a new report, named MailMerge
3. Do not check the option for "Use Report Wizard"

Set the Page Layout

1. Select "File" > "Page Setup" from the Report Designer main Menu Bar.
2. Select the "Margins" tab.
3. Set all margins to 1.25.
4. Click the "OK" button to close the Page Setup dialog.

Modify/Remove the Report Bands

1. From the main Menu Bar, select "Report".
2. Within the submenu options, uncheck the Report Header option.
3. Within the submenu options, uncheck the Report Footer option.
4. Within the submenu options, uncheck the Page Header option.
5. Within the submenu options, uncheck the Page Footer option.
6. Right-click over the white space of the Detail band and select "Position..." from the speed menu
7. Set the "Height" to **4** and "Print Count" to **1**. This will allow only one detail band to print per page, thus creating the effect of one form letter per page.
8. Select "Apply" and then the "OK" button to save the Position parameters.

The Advanced Rich Text Control

1. Place an Advanced Rich Text control (part of Standard Controls) in the Detail band.
2. Right-click over the Advanced Rich Text control and select the “Stretch” menu option. This will force the control to resize based on the size of the control contents.
3. Right-click over the Advanced Rich Text control and set the position and size as follows:

   Left: 0
   Top: 0
   Width: 6
   Height: 4

4. Select "Apply" and then the "OK" button to save the Position parameters.

The Advanced Rich Text control is positioned so that it fills the entire detail band. This allows us to use the margins of the report to control the positioning of the letter, as opposed to positioning the Advanced Rich Text control within the band. The height of 4 is arbitrary; it will simply allow us to read the entire contents of the letter while designing. When the report prints, the Advanced Rich Text control will calculate its height based on the length of the letter.

Enter the Body of the Letter into the Advanced Rich Text Control

1. Right-click over the Advanced Rich Text control and select the "Mail Merge" menu option.
2. Right-click over the Advanced Rich Text control and select the "Edit" menu option. The Advanced Rich Text Editor will be displayed.
3. Type or copy and paste the body of your mail merge document in the Advanced Rich Text Editor, leaving a few empty lines at the top for your customer name, address, etc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Add Table Fields to the Letter

1. From within the Advanced Rich Text Editor, select "Insert" > "DB Label" from the Menu Bar. A list of columns from the table will be displayed. After each field is added, you will be prompted to add a "Format Mask", which is optional.
2. Select the appropriate address heading fields starting with the addressee first and last name. Each field must be added separately. Be sure to add the spaces between first name and last name fields.
3. Then press the "Enter" key to skip to the next line and add the company DB Label.
4. Press the "Enter" key and add the address DB Label(s).
5. Now, press the "Enter" key and add the city, state, and zip code DB Labels. Be sure to place a comma and a space between the city and state, and a space between the state and zip code.
6. Press the "Enter" key twice to create a blank line below the address fields.
7. Type "Dear", then a space, then insert the first name DB Label.
8. Type a comma and press the "Enter" twice, leaving a blank line between the letter greeting and the body added earlier.
9. Press the "OK" button to save your work and close the Advanced Rich Text Editor.

Preview the Report in Report Designer

1. Click on the "Preview" tab n the Report Designer to preview the report. If any changes are required make them now.
2. Save the report and close Report Designer.
3. At the R> Prompt, type:

```
PRINT MailMerge OPTION SCREEN|WINDOW_STATE MAXIMIZED
```

Notes:

- Use the Advanced Rich Text Editor to update the body of the letter, by right clicking over the Advanced Rich Text control and selecting the "Edit" menu option.
- Column fields are enclosed within less than (<)and greater than (>) characters.
- Variable fields are enclosed within braces {}.
- Use the "Insert" > "Variable Label" menu option to insert variable expressions or system variables, such as #DATE or #TIME.
- You can save the rich text as an external RTF file.
- Any Rich Text data created with your favorite word processor can easily be copied into the Advanced Rich Text Editor.
- To insert the field or variable, you may also use right-click in the body of the letter for speed menu options.

1.42 Using the Data Dictionary

The Data Dictionary is an R:BASE program utility that displays, and allows users to capture, information about the R:BASE environment and connected database.

The Data Dictionary is accessible from anywhere in R:BASE using the following methods:

1. Pressing the [F3] key
2. Selecting "Tools" > "Data Dictionary" from the main menu bar
3. Clicking on the Data Dictionary button, where provided

When using the Data Dictionary to capture information, such as a table name or several columns, the selection is automatically inserted where the cursor is located within an available input, e.g. the R:BASE Editor, or a text file. The "Apply" button captures and inserts the selected information and the window remains open. Clicking "OK" will capture the selected information and the window will close. If no input is available, the selection is stored in the Windows clipboard. If in the clipboard, the selection will be provided by pasting the content using [Ctrl+V] or by using a "Paste" menu option. When selecting multiple items within a list, the values captured are separated by a comma (or the current delimiter).

Tabs

The Data Dictionary tabs are organized with distinctive color identifiers (triangle in top right corner) to improve the readability for the several different types of information presented. The tab stacking order may also be reorganized with drag-and-drop of an individual tab. The [Ctrl+Page Down] and [Ctrl+Page

Copyright © 1982-2024 R:BASE Technologies, Inc.

Up] hot keys allow the next/previous tab navigation. Whenever the Data Dictionary is launched, the display will default to the last tab used to capture information.

Based upon where the Data Dictionary is launched within R:BASE, the tabs displayed will vary. There are tabs which are only displayed while connected to a database, others that are only displayed while the Data Dictionary is launched within a designer interface, and other tabs that are permanently displayed.

- Permanent Tabs
- Database Tabs
- Designer Tabs

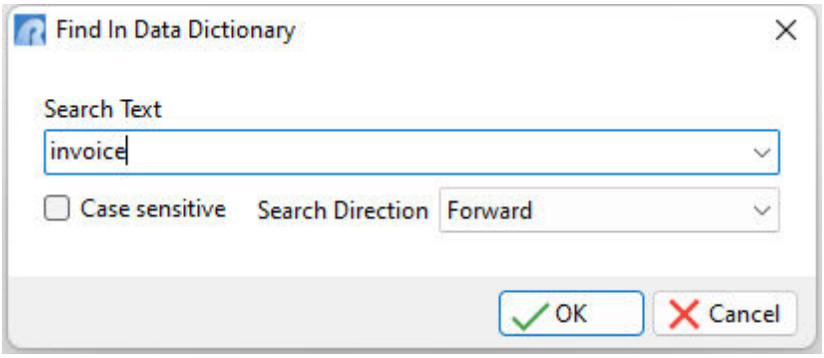
MDI
The Data Dictionary recognizes the MDI setting, and when ON, will remain open while other areas of R:BASE are used. When MDI is ON, the "OK" button is not displayed. When focus is switched back to the Data Dictionary, the content will automatically refresh (only in MDI mode).

Buttons
The Data Dictionary buttons alter the display of items listed within the window, how the tabs are displayed, and how the selected values are captured.

Icon	Function
	Displays items as large icons
	Displays items as small icons
	Displays items as a list
	Displays items as details
	Displays tabs on the top edge
	Displays tabs on the bottom edge
	Displays tabs on the left edge
	Displays tabs on the right edge
	Includes a space directly after the comma, so that after a string of values is added to a command file, the [Ctrl+Arrow] keys can be used to move the cursor between values with greater ease
	Show/Hide Grid Lines
	Show/Hide Multi-line Tabs

Text Searches
A "Find in Data Dictionary" search utility [Ctrl+N] is available to locate text of objects within the active tab. The entry box is also a drop down where the search history is available in the drop down items. To clear the history, press the [Ctrl+R] key combination. The search history is saved in the C:\Users\<User>\RBTI\ folder within the DATADICT.RST file.

Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.1 Permanent Tabs

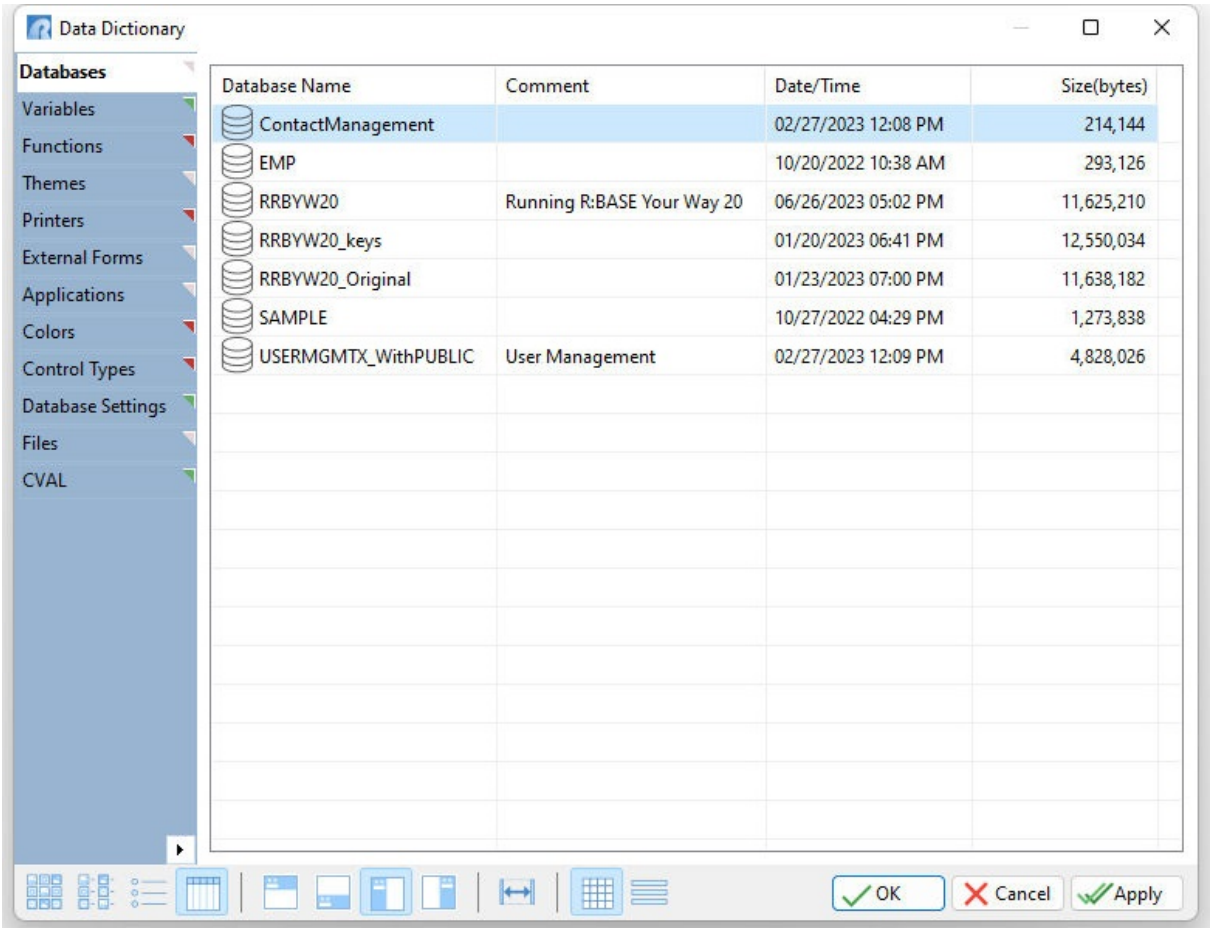
There are permanent tabs which are always displayed when the Data Dictionary is launched. The Databases, External Forms, Files, and Applications tabs list external file resources, which are available for the current folder for the R:BASE instance. Changing the current folder for the instance of R:BASE will alter the items displayed within these tabs.

- Databases
- Variables
- Functions
- Themes
- Printers
- External Forms
- Applications
- Colors
- Control Types
- Database Settings
- Files
- CVAL

1.42.1.1 Databases

The Databases tab lists all databases in the current folder. The database name, comment, date/time stamp for the files, and size is provided. With a database selected, pressing the OK button will capture the database name.

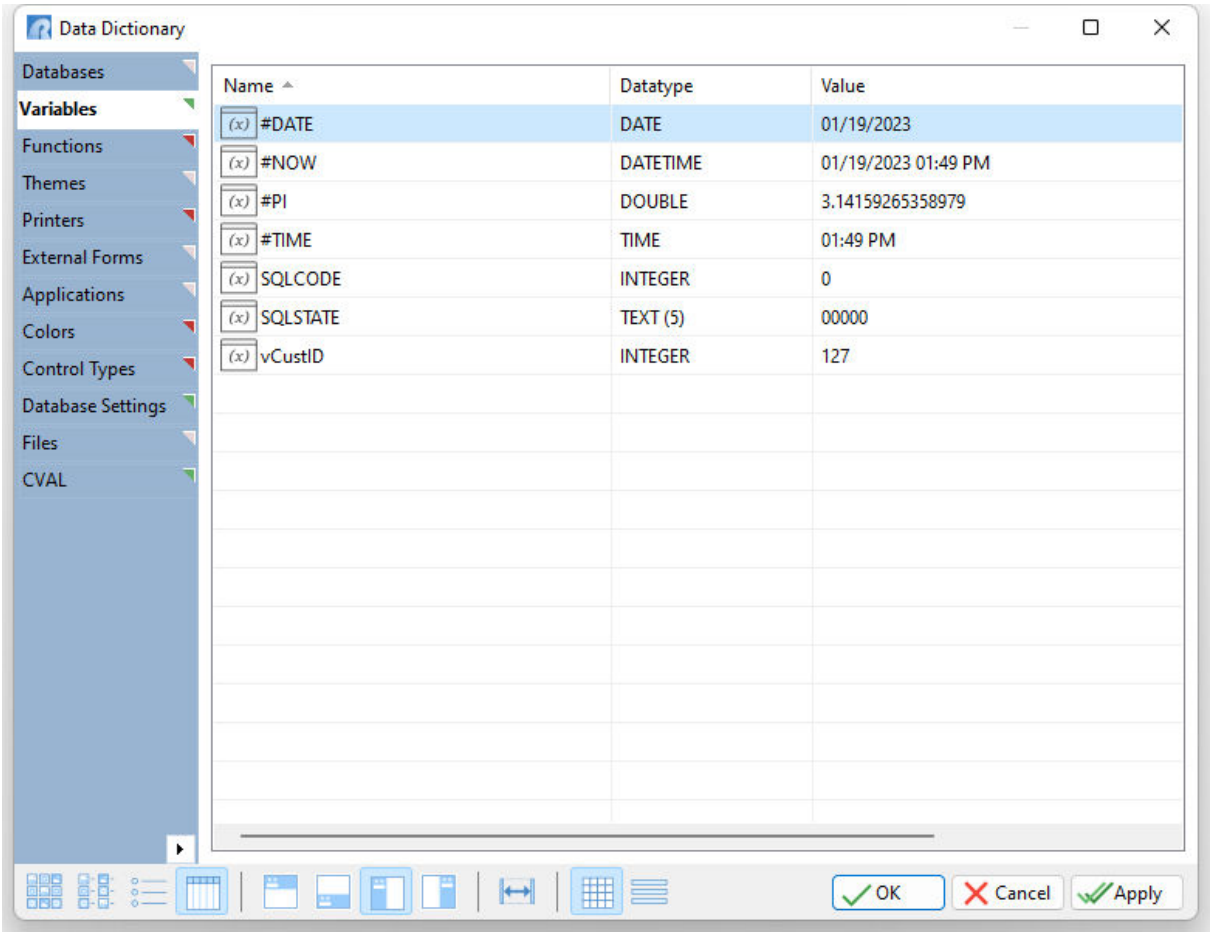
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.1.2 Variables

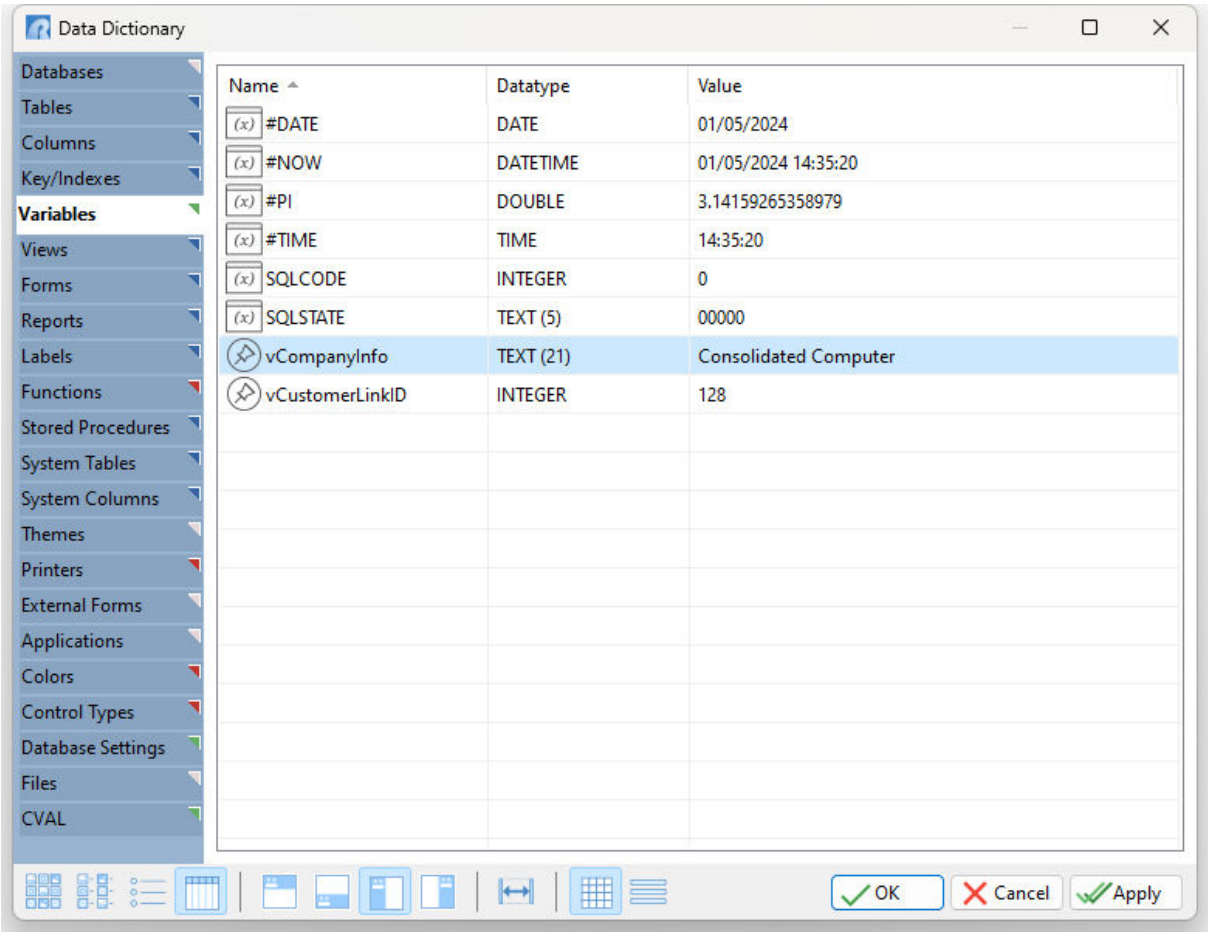
The Functions tab lists all available predefined functions supported by the current R:BASE version. The function name, category, syntax, and description is provided. With a function selected, pressing the OK button will capture the function syntax. A drop-down box is available to filter the functions by category.

Copyright © 1982-2024 R:BASE Technologies, Inc.



When static variables are defined, the variables will display with a push pin icon.

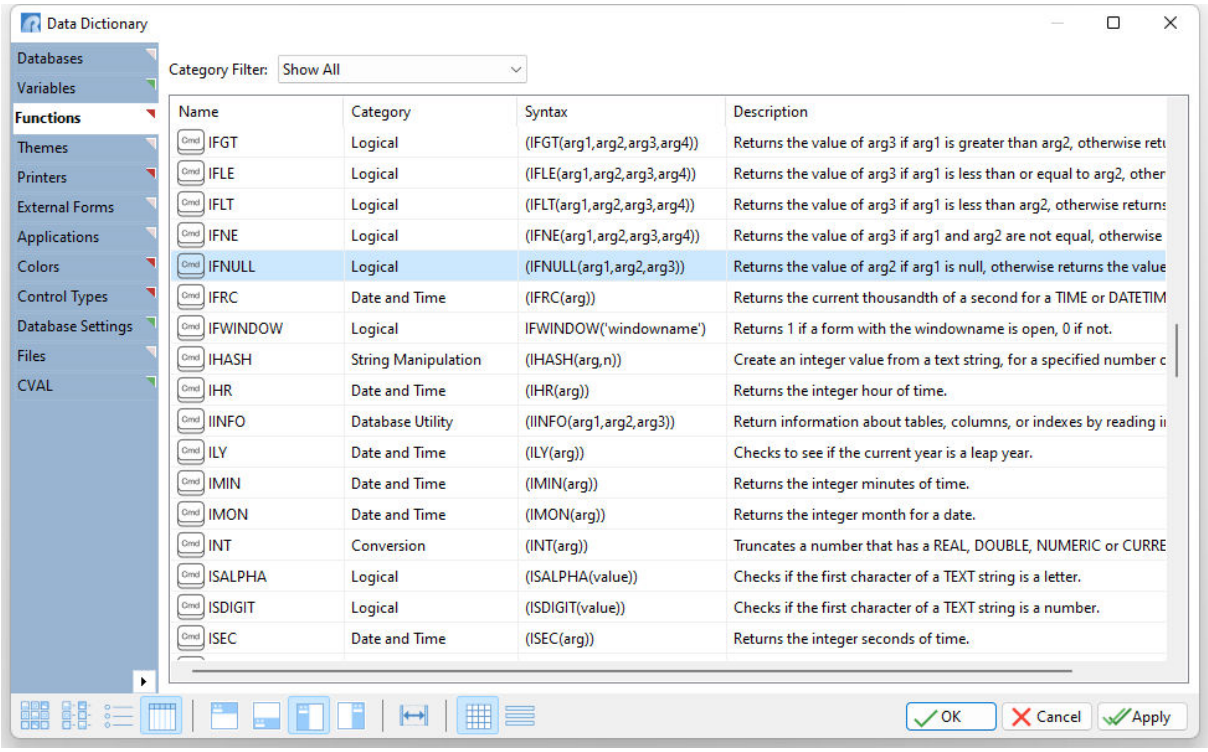
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.1.3 Functions

The Functions tab lists all available predefined functions supported by the current R:BASE version. The function name, category, syntax, and description is provided. With a function selected, pressing the OK button will capture the function syntax. A drop-down box is available to filter the functions by category.

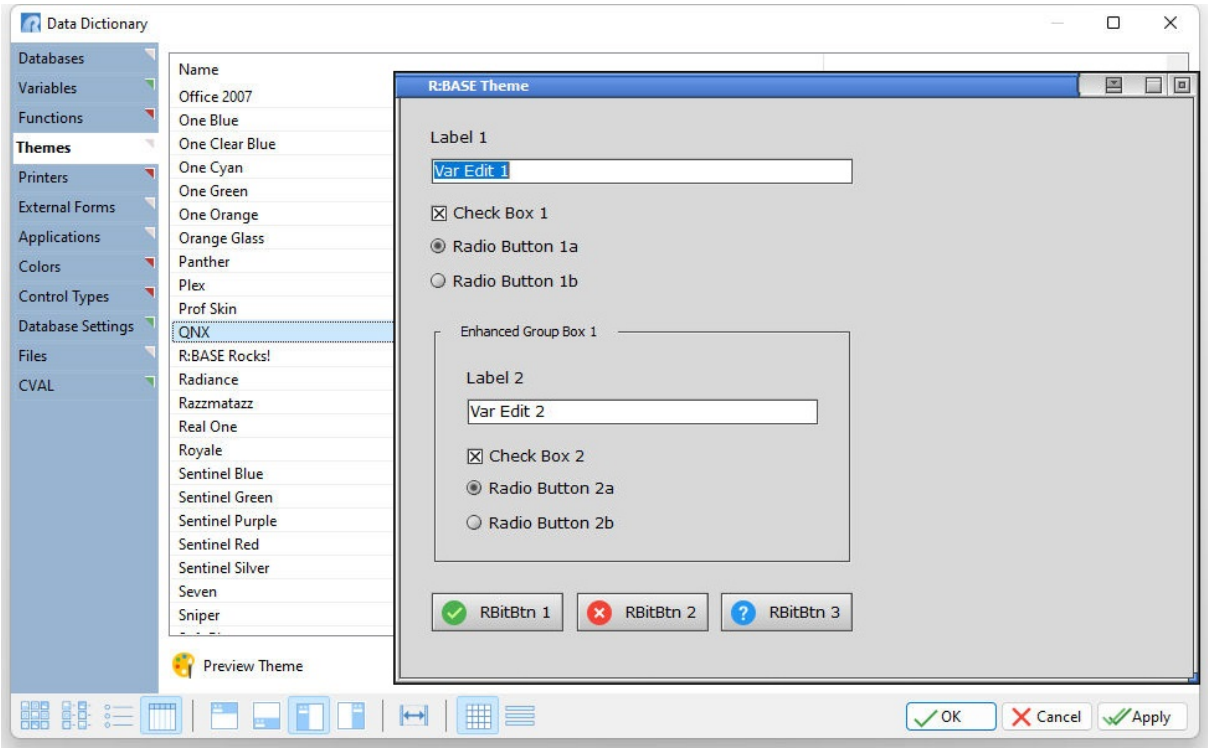
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.1.4 Themes

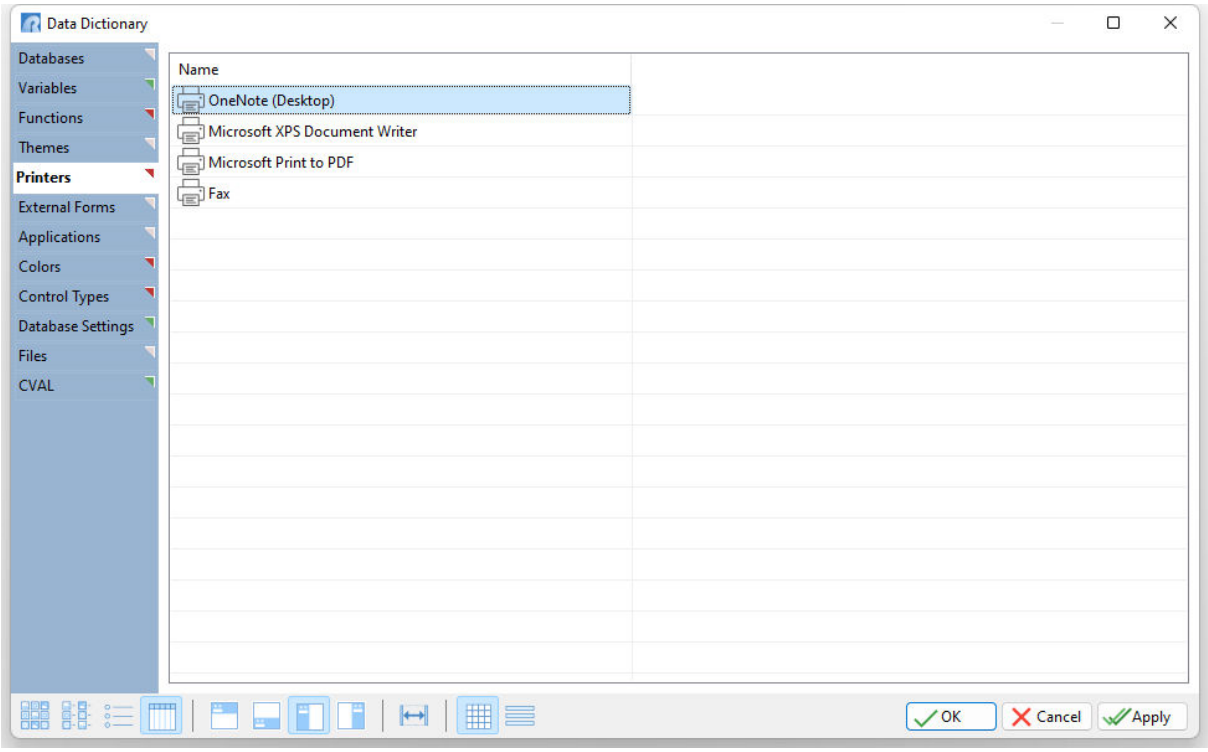
The Themes tab lists all available themes loaded into the R:BASE instance. The theme list includes the 86 predefined themes available in R:BASE, and any external themes that were loaded into R:BASE. With a theme selected, pressing the OK button will capture the theme name. The RBThemes11.dll must be located in the R:BASE program folder in order for the themes tab to be displayed. The "Preview Theme" button provides a sample form to review the theme presentation.

Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.1.5 Printers

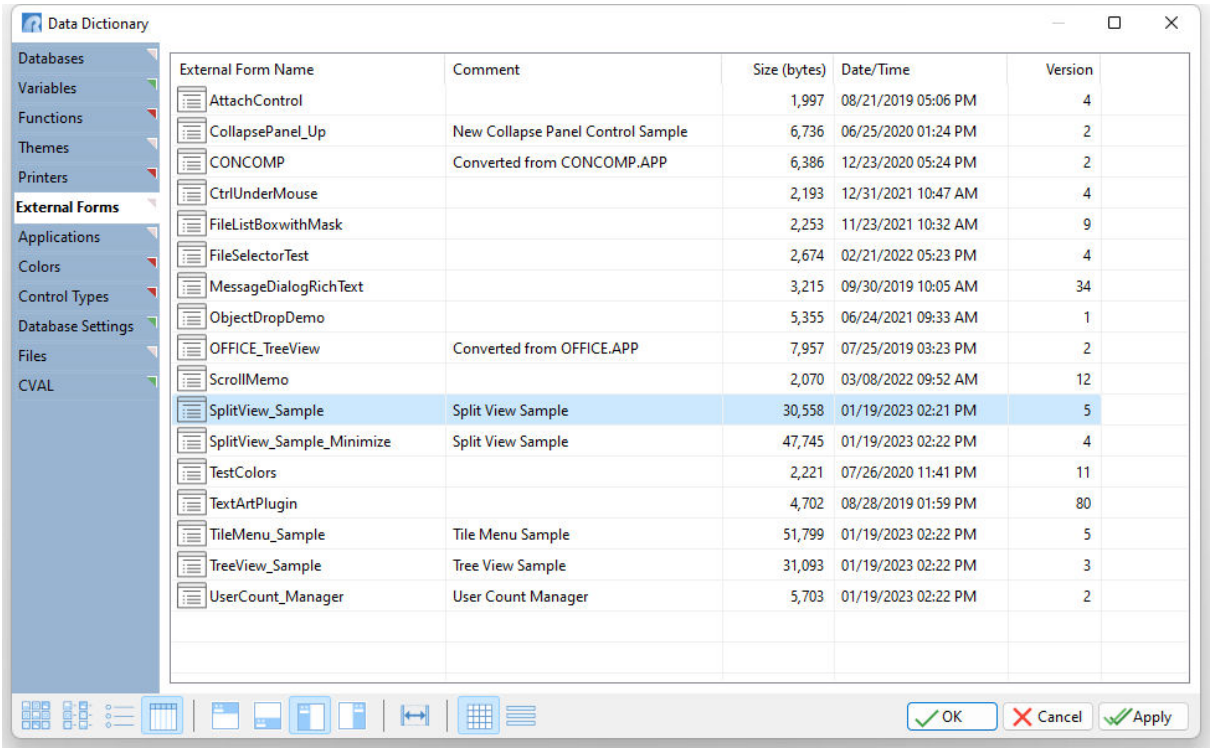
The Printers tab lists all available printers installed for the current operating system. With a printer selected, pressing the OK button will capture the printer name.



Copyright © 1982-2024 R:BASE Technologies, Inc.

1.42.1.6 External Forms

The External Forms tab lists all external form files (.rff) in the current folder. The external form name, comment, size, date/time stamp, and version is provided. With an external form file selected, pressing the OK button will capture the external form file name.

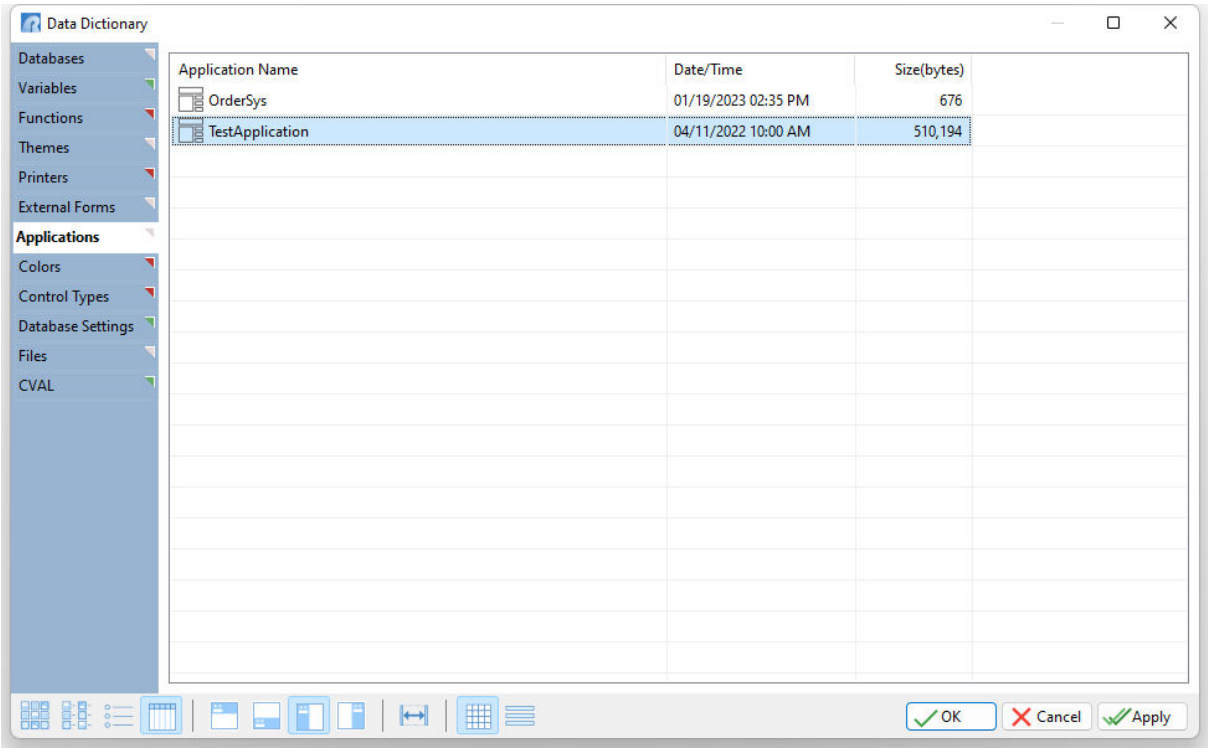


External Form Name	Comment	Size (bytes)	Date/Time	Version
AttachControl		1,997	08/21/2019 05:06 PM	4
CollapsePanel_Up	New Collapse Panel Control Sample	6,736	06/25/2020 01:24 PM	2
CONCOMP	Converted from CONCOMP.APP	6,386	12/23/2020 05:24 PM	2
CtrlUnderMouse		2,193	12/31/2021 10:47 AM	4
FileListBoxwithMask		2,253	11/23/2021 10:32 AM	9
FileSelectorTest		2,674	02/21/2022 05:23 PM	4
MessageDialogRichText		3,215	09/30/2019 10:05 AM	34
ObjectDropDemo		5,355	06/24/2021 09:33 AM	1
OFFICE_TreeView	Converted from OFFICE.APP	7,957	07/25/2019 03:23 PM	2
ScrollMemo		2,070	03/08/2022 09:52 AM	12
SplitView_Sample	Split View Sample	30,558	01/19/2023 02:21 PM	5
SplitView_Sample_Minimize	Split View Sample	47,745	01/19/2023 02:22 PM	4
TestColors		2,221	07/26/2020 11:41 PM	11
TextArtPlugin		4,702	08/28/2019 01:59 PM	80
TileMenu_Sample	Tile Menu Sample	51,799	01/19/2023 02:22 PM	5
TreeView_Sample	Tree View Sample	31,093	01/19/2023 02:22 PM	3
UserCount_Manager	User Count Manager	5,703	01/19/2023 02:22 PM	2

1.42.1.7 Applications

The Applications tab lists all application files (.rba) in the current folder. The application file name, date/time stamp, and size is provided. With an application file selected, pressing the OK button will capture the application file name.

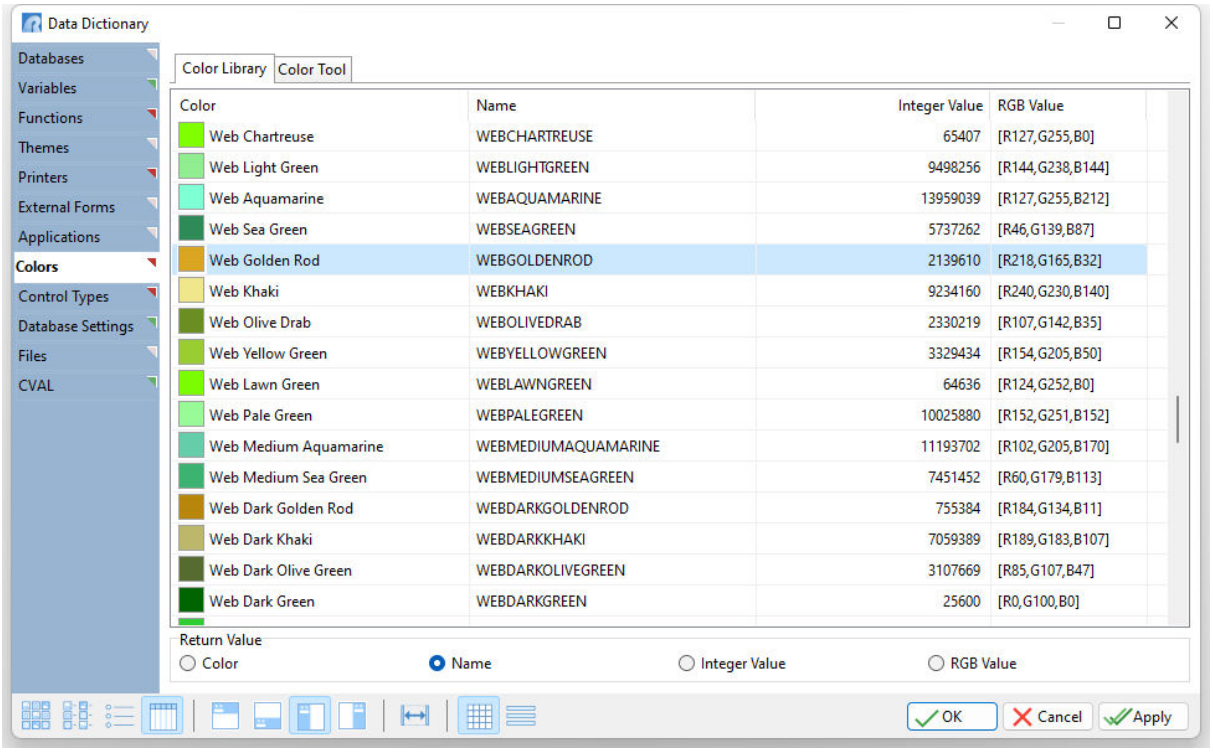
Copyright © 1982-2024 R:BASE Technologies, Inc.



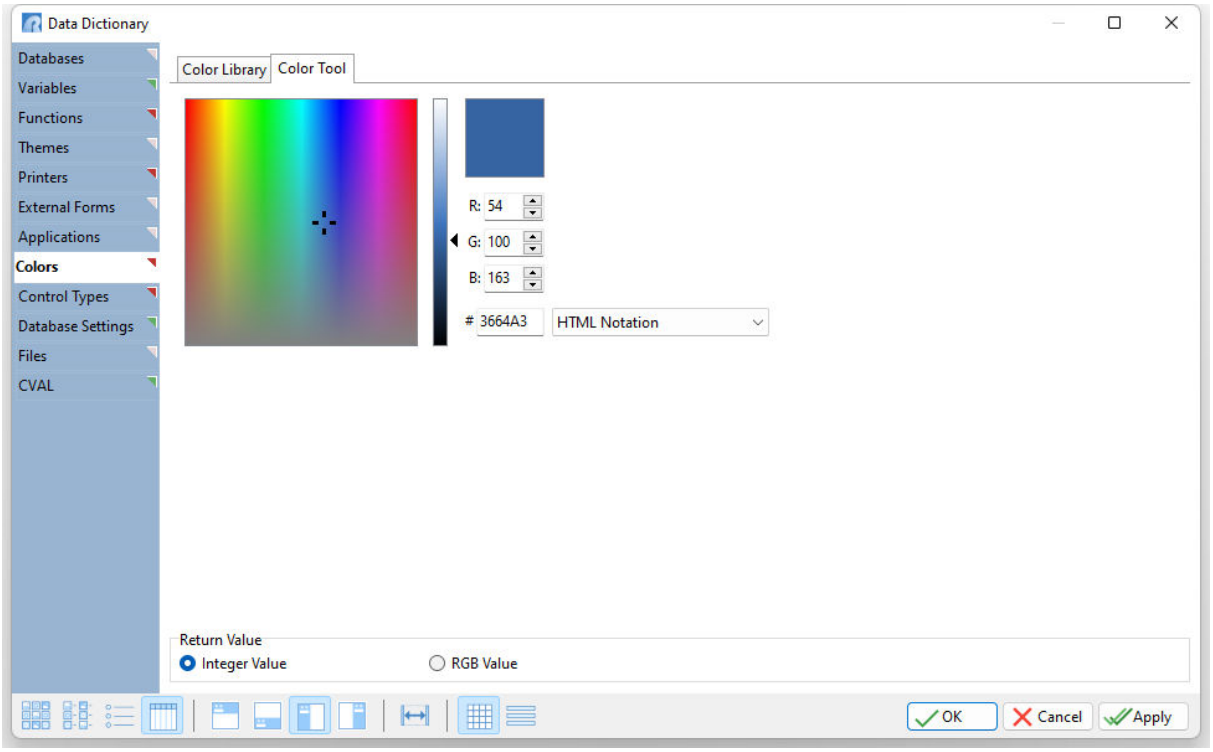
1.42.1.8 Colors

The Colors tab lists the available 190 colors supported by R:BASE. The color name, integer value, and RGB value is provided. Select one of the available radio button options (Color, Name, Integer Value, RGB Value) within the Return Value panel. With the "Name" radio button selected and by choosing the desired color and the OK button, the color name will be captured, e.g. NAVY. With the "RGB Value" radio button selected, and by choosing the desired color and the OK button, the RGB value will be captured, e.g. [R0,G255,B127].

Copyright © 1982-2024 R:BASE Technologies, Inc.



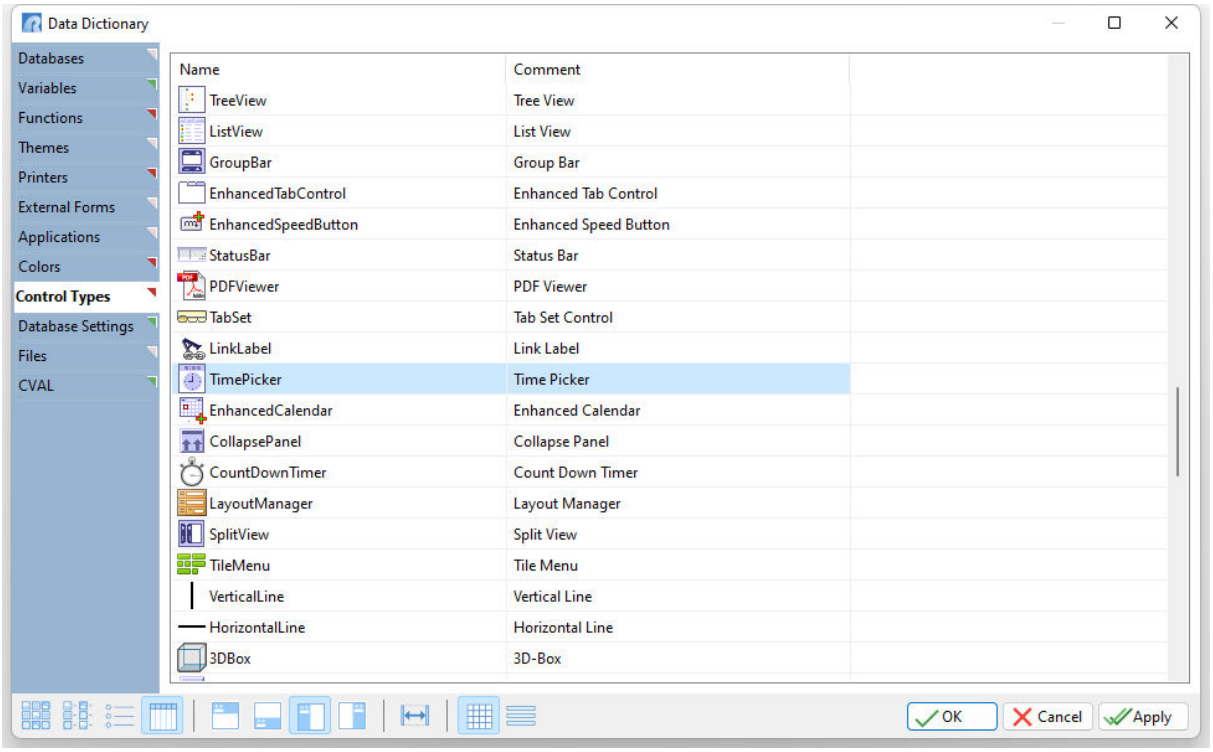
The "Color Tool" tab provides a mechanism to perform color value conversions.



Copyright © 1982-2024 R:BASE Technologies, Inc.

1.42.1.9 Control Types

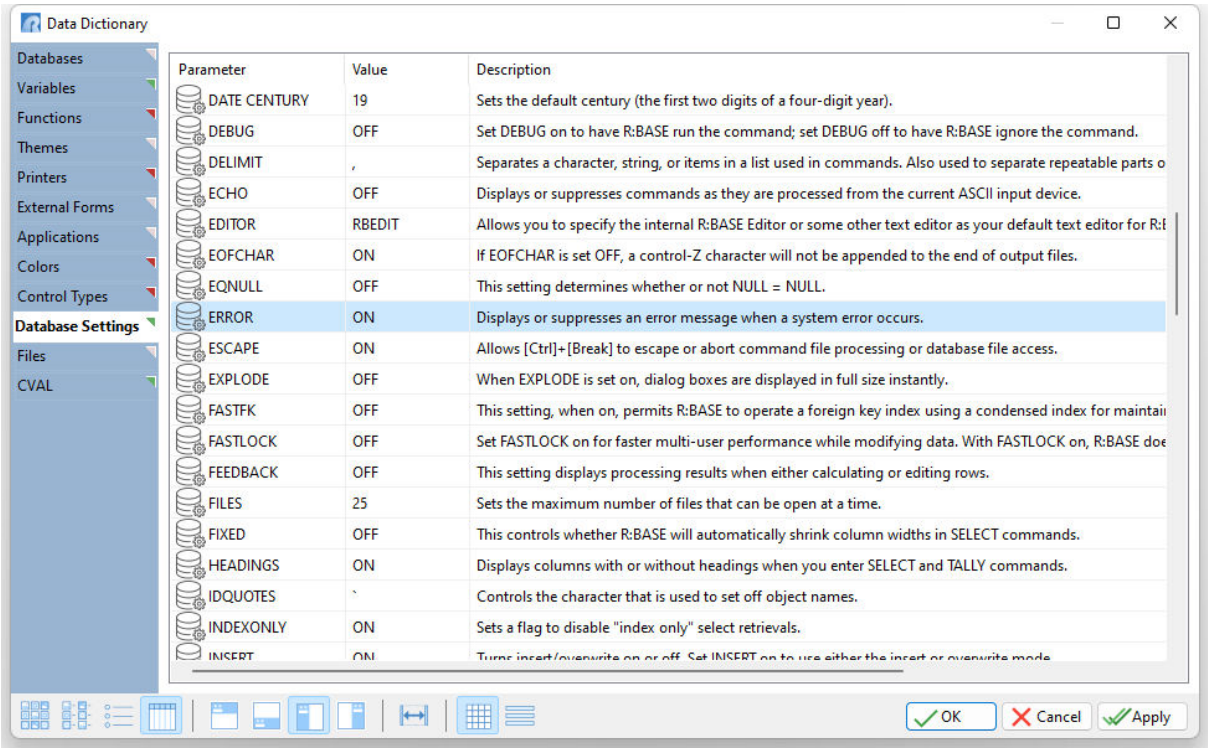
The Control Types tab lists all available form controls supported by the current R:BASE version. The control type name and comment is provided. With a control type selected, pressing the OK button will capture the control type name. The Control Types tab is useful when creating form objects on demand with the CREATEOBJECT command.



1.42.1.10 Database Settings

The Database Settings tab lists all available database settings supported by the current version of R:BASE. The database setting parameter name, value, and description is provided. With a database settings selected, pressing the OK button will capture the parameter name.

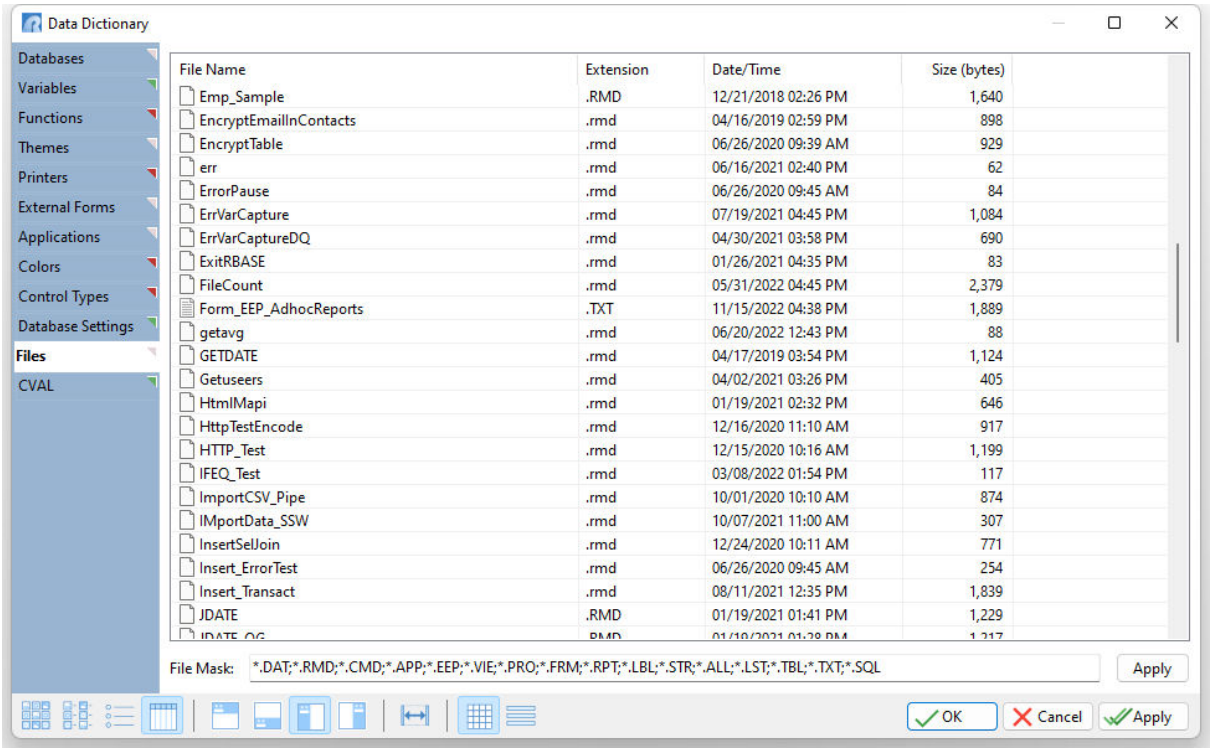
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.1.11 Files

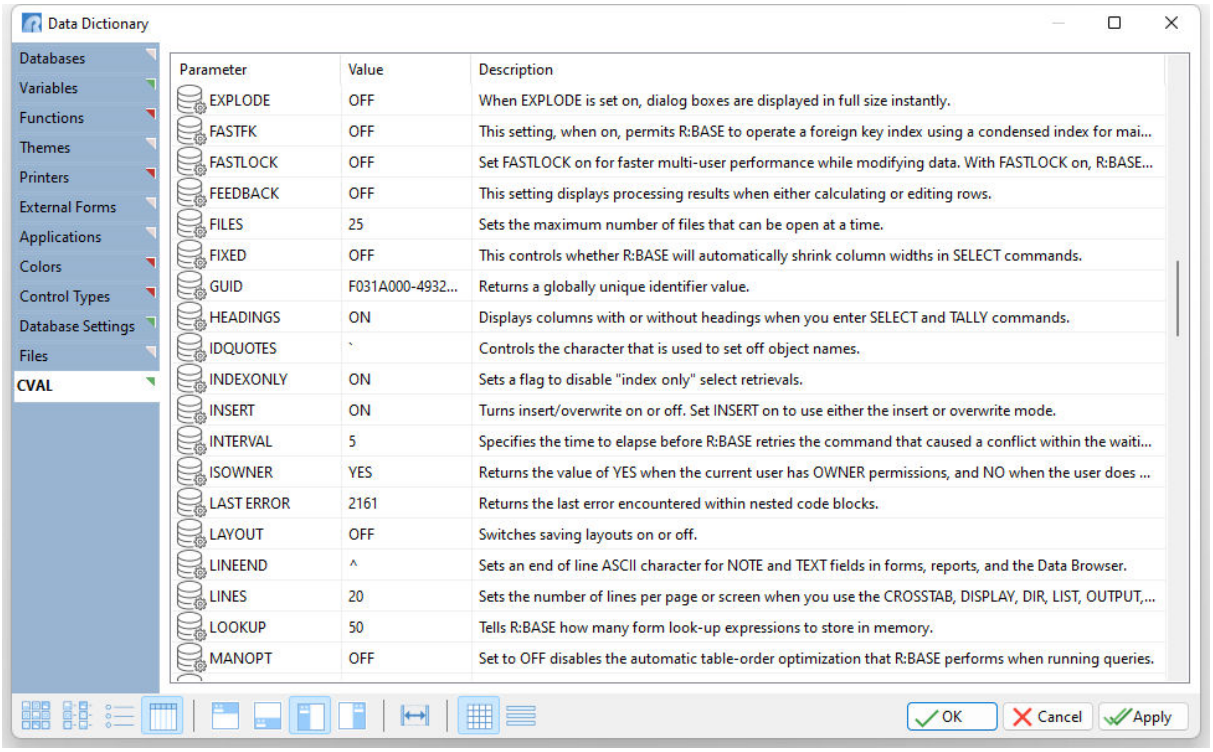
The Files tab lists all file in the current folder. The file name, extension, date/time stamp, and size is provided. With a file selected, pressing the OK button will capture the file name. The "File Mask" field filters specific file extensions that are to be displayed. All values are semi-colon (;) separated, and are in the "*.EXT" format. The file icons displayed are based upon the Windows programs associated to those file extensions.

Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.1.12 CVAL

The CVAL tab lists all CVAL Functions to review the current value, and to easily add the function to command syntax.



Copyright © 1982-2024 R:BASE Technologies, Inc.

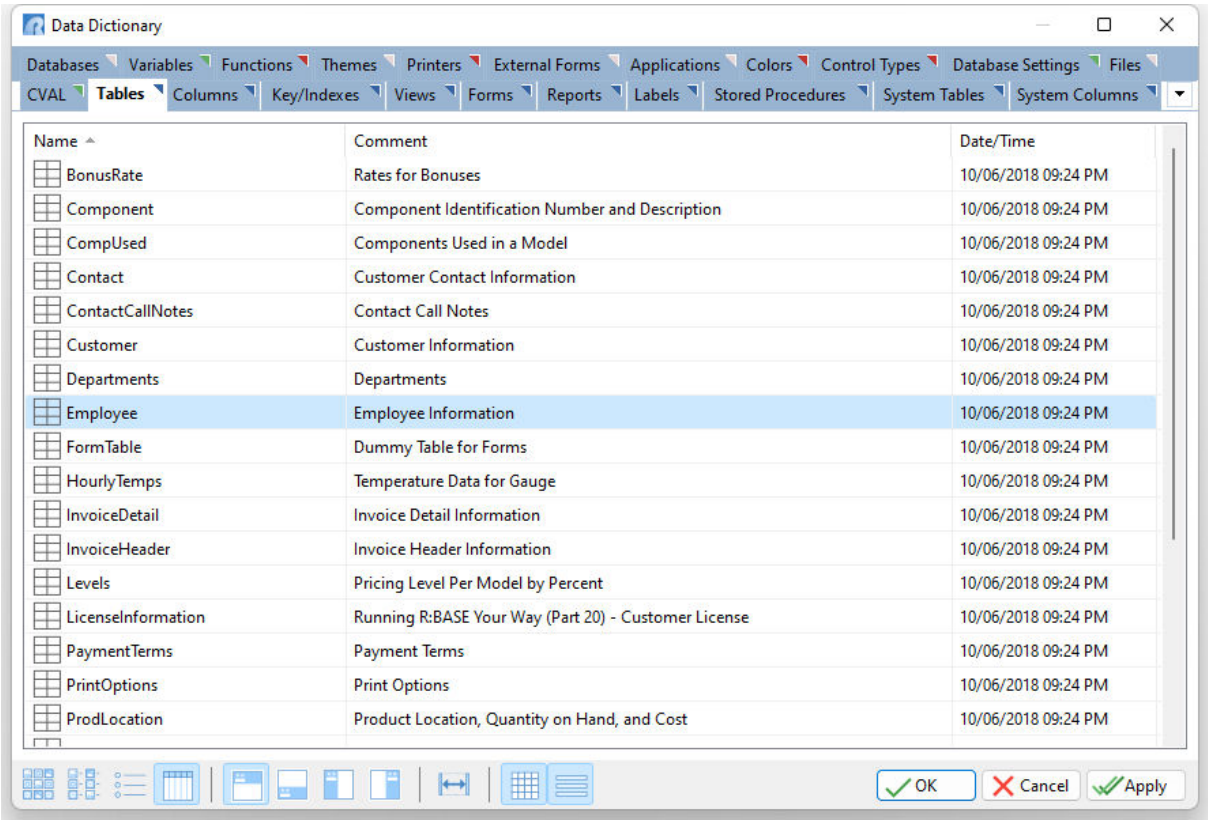
1.42.2 Database Tabs

The database tabs are displayed when the Data Dictionary is launched while R:BASE is connected to a database.

- Tables
- Columns
- Keys/Indexes
- Views
- Forms
- Reports
- Labels
- stored procedures
- System Tables
- System Columns

1.42.2.1 Tables

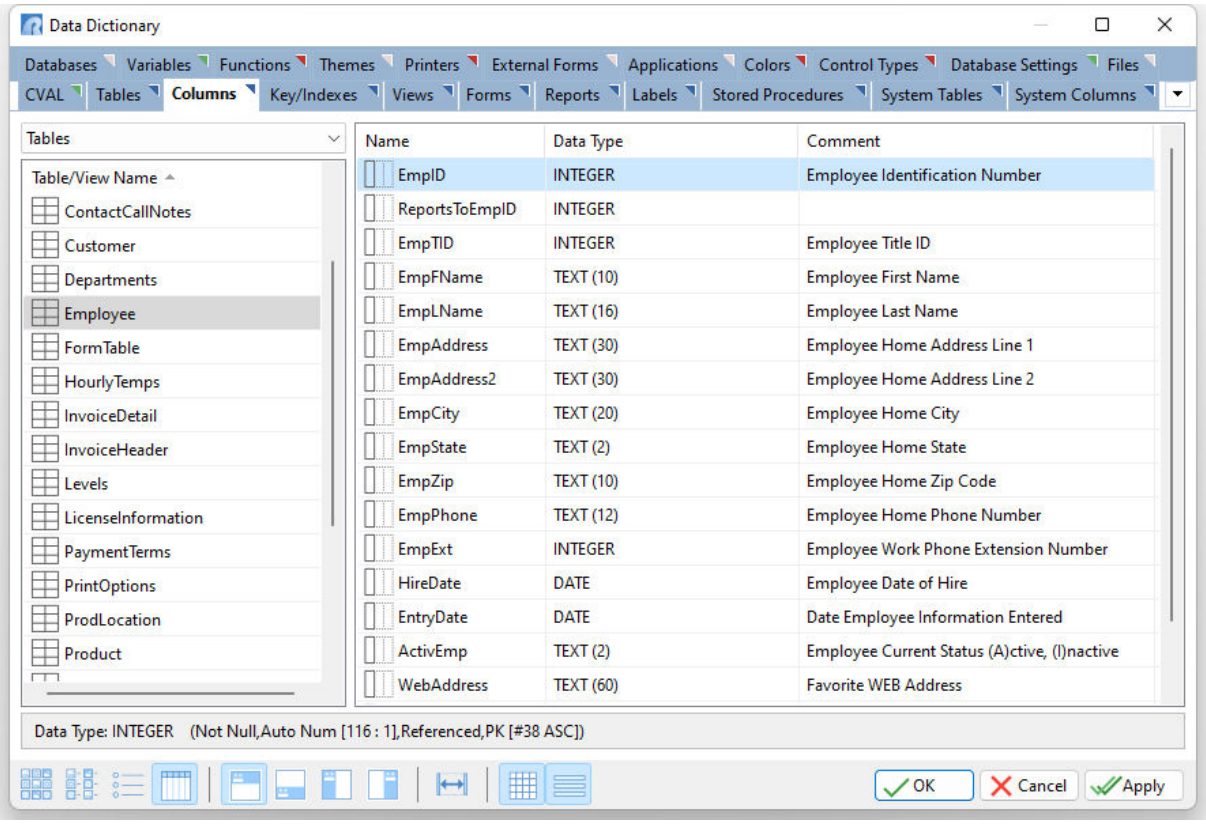
The Tables tab lists all defined tables, temporary tables, server tables, and dBASE tables in the current database. The table name, comment, and last modified date/time for the structure is provided. With a table selected, pressing the OK button will capture the table name.



1.42.2.2 Columns

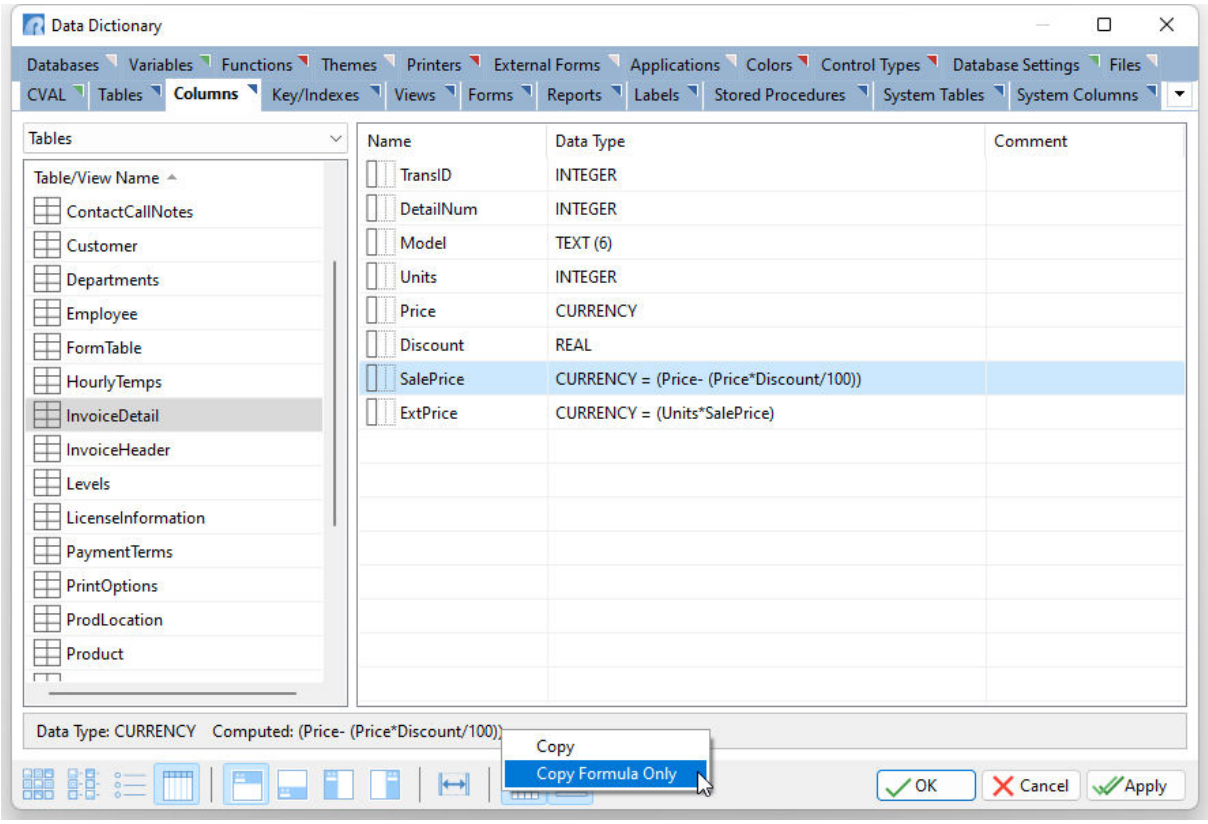
The Columns tab lists all columns with the defined tables and views in the current database. A drop-down box is available to browse the list of tables and views, only tables, or only views. The column name, data type, and comment is provided. With a column selected, the data type and column definition (indexed, primary key, foreign key, computed, autonum, not NULL, order, etc.) is provided. Pressing the OK button will capture the column name. When selecting multiple columns, the values captured are separated by a comma (or the current delimiter).

Copyright © 1982-2024 R:BASE Technologies, Inc.



When a computed column is displayed in the Data Dictionary, the ability to copy the entire column definition or just the formula is available by right clicking on the displayed computed column calculation in the Data Dictionary window. If you wish to retain the copied formula in the clipboard, select the "Cancel" button and not the"OK" button. Choosing the "OK" button will overwrite the formula string in the clipboard with currently selected column(s).

Copyright © 1982-2024 R:BASE Technologies, Inc.

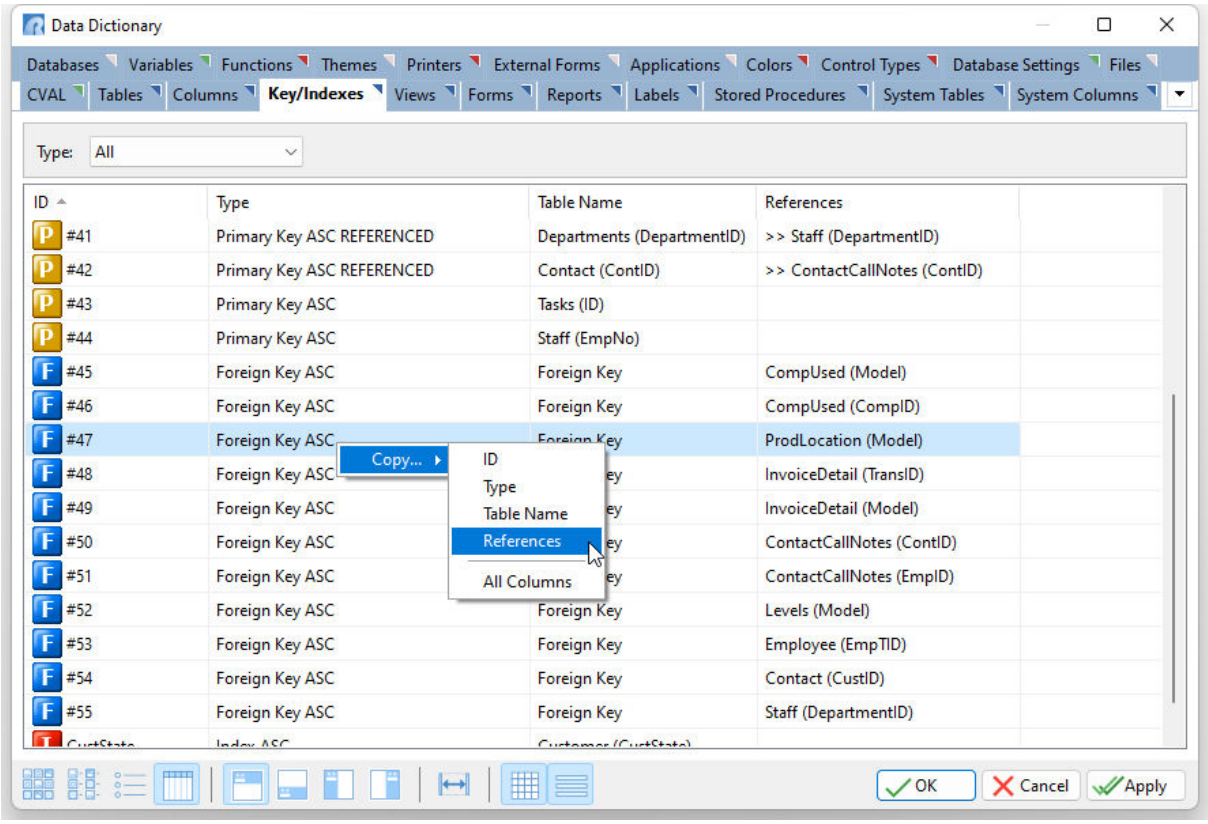


1.42.2.3 Keys/Indexes

The Key/Indexes tab lists all defined primary keys, foreign keys, unique keys, and indexes in the current database. With a key/index selected, pressing the OK button will capture the key/index name. A drop-down box is available to browse the keys/indexes by type. Selecting "Foreign Keys" from the drop-down box will only display the list of foreign keys.

The ability to copy the ID, Type, Table Name, References, or all columns for the selected key/index is available by right clicking on the displayed key/index in the Data Dictionary window. If you wish to retain the copied values in the clipboard, select the "Cancel" button and not the"OK" button. Choosing the "OK" button will overwrite the value in the clipboard with currently selected ID.

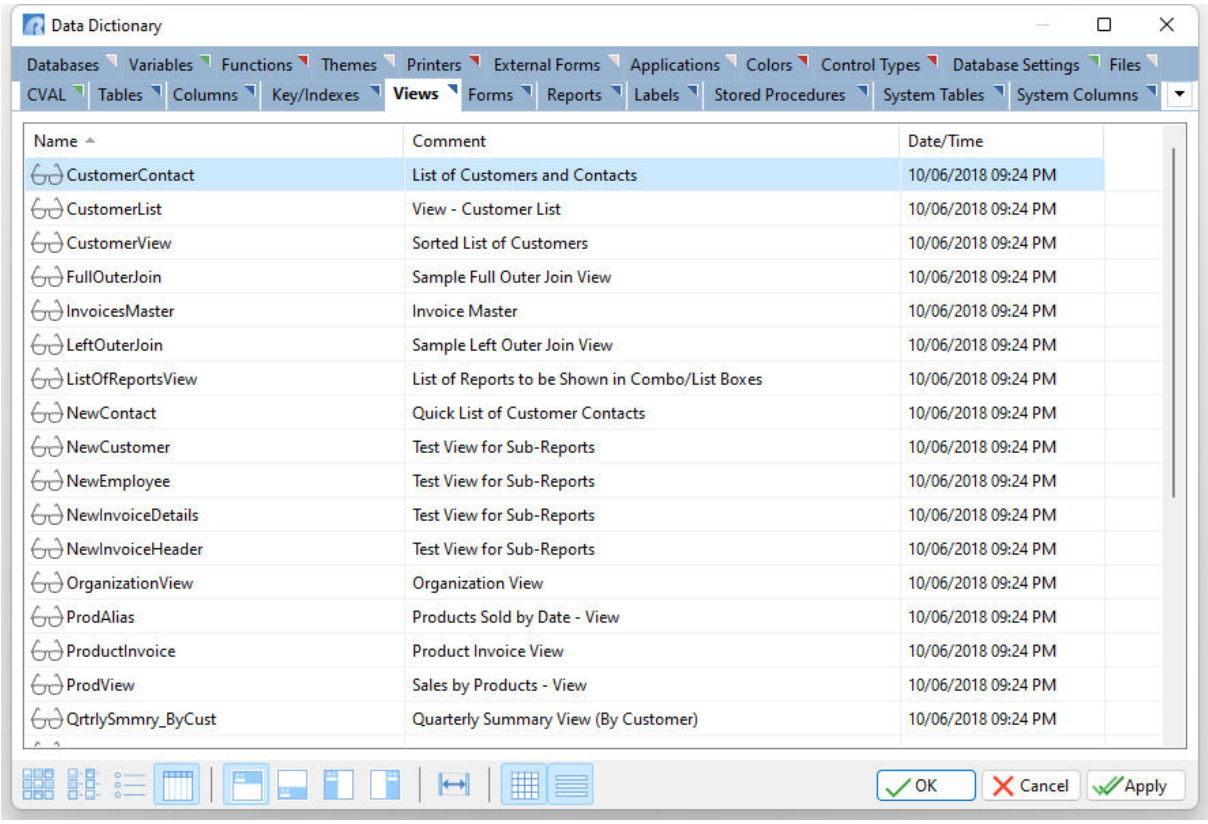
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.4 Views

The Views tab lists all defined views in the current database. The view name, comment, and last modified date/time for the structure is provided. With a view selected, pressing the OK button will capture the view name.

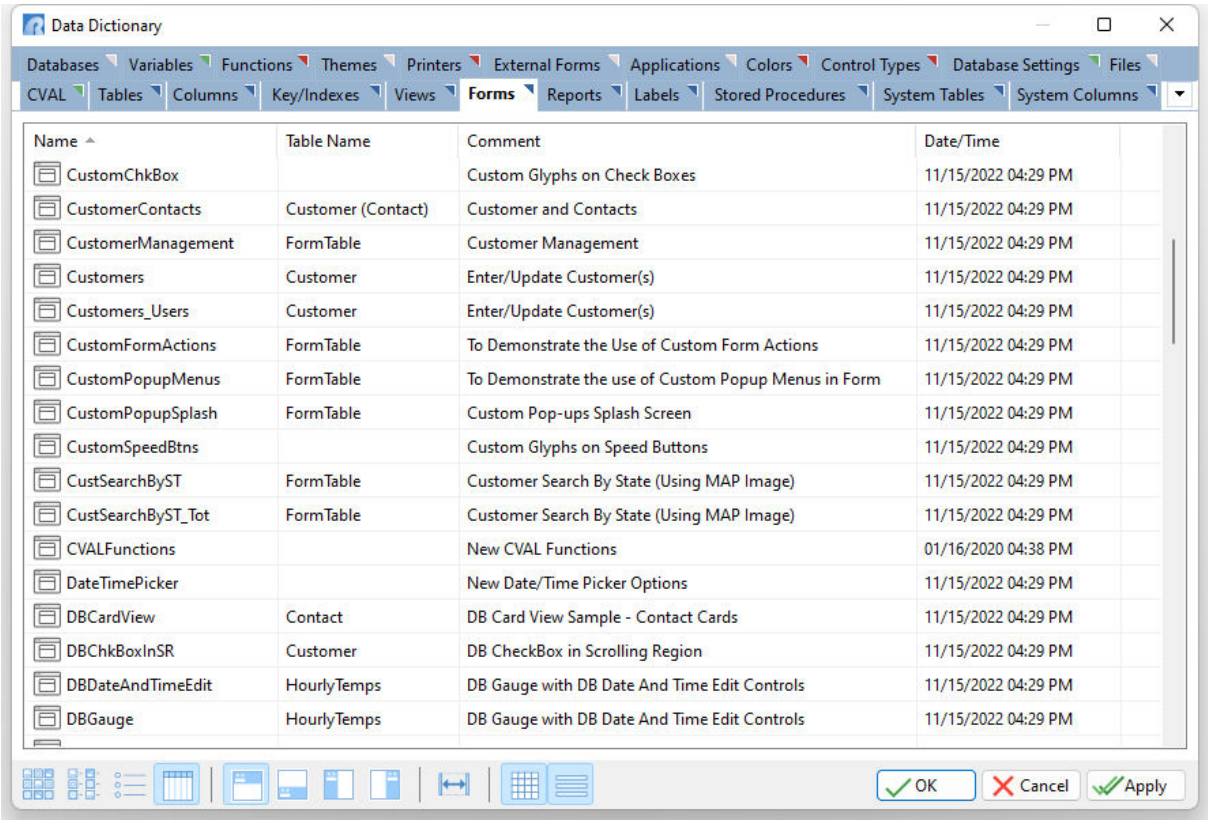
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.5 Forms

The Forms tab lists all defined forms in the current database. The form name, table name, comment, and last modified date/time is provided. With a form selected, pressing the OK button will capture the form name.

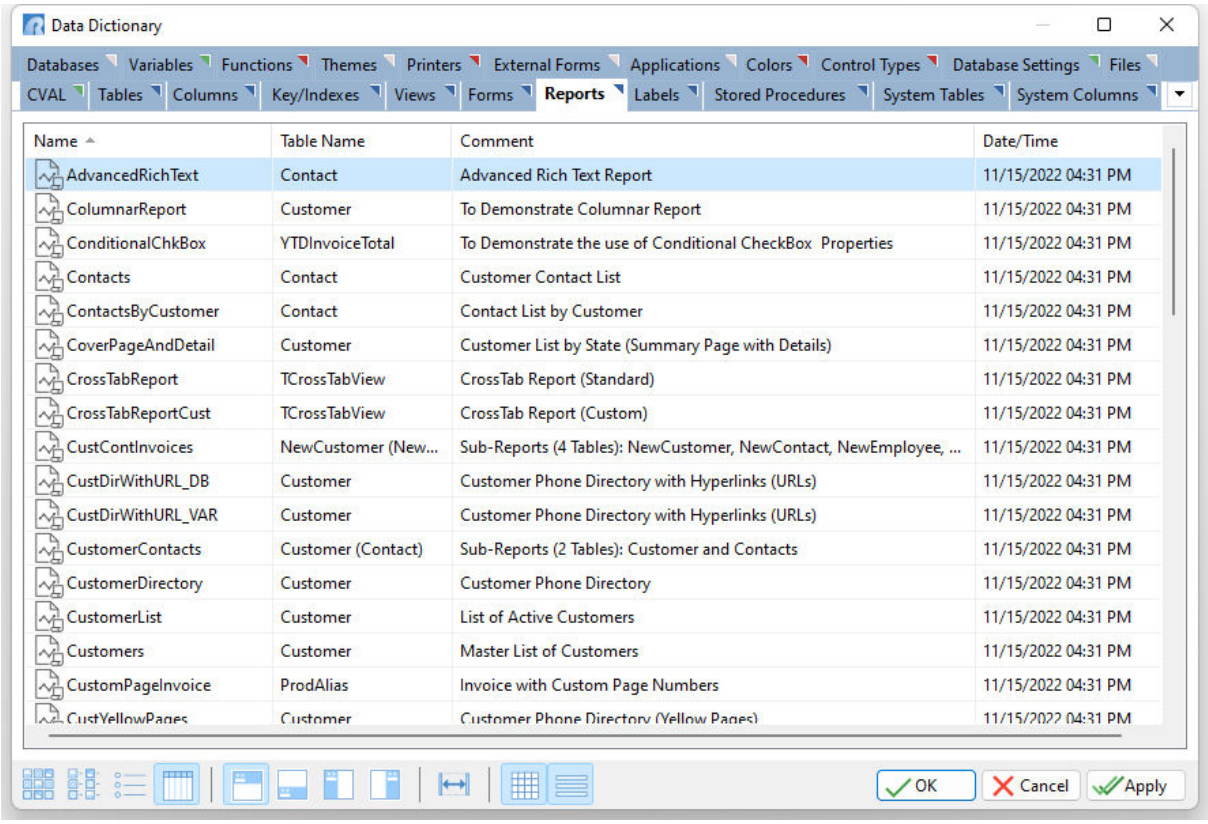
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.6 Reports

The Reports tab lists all defined reports in the current database. The report name, table name, comment, and last modified date/time is provided. With a report selected, pressing the OK button will capture the report name.

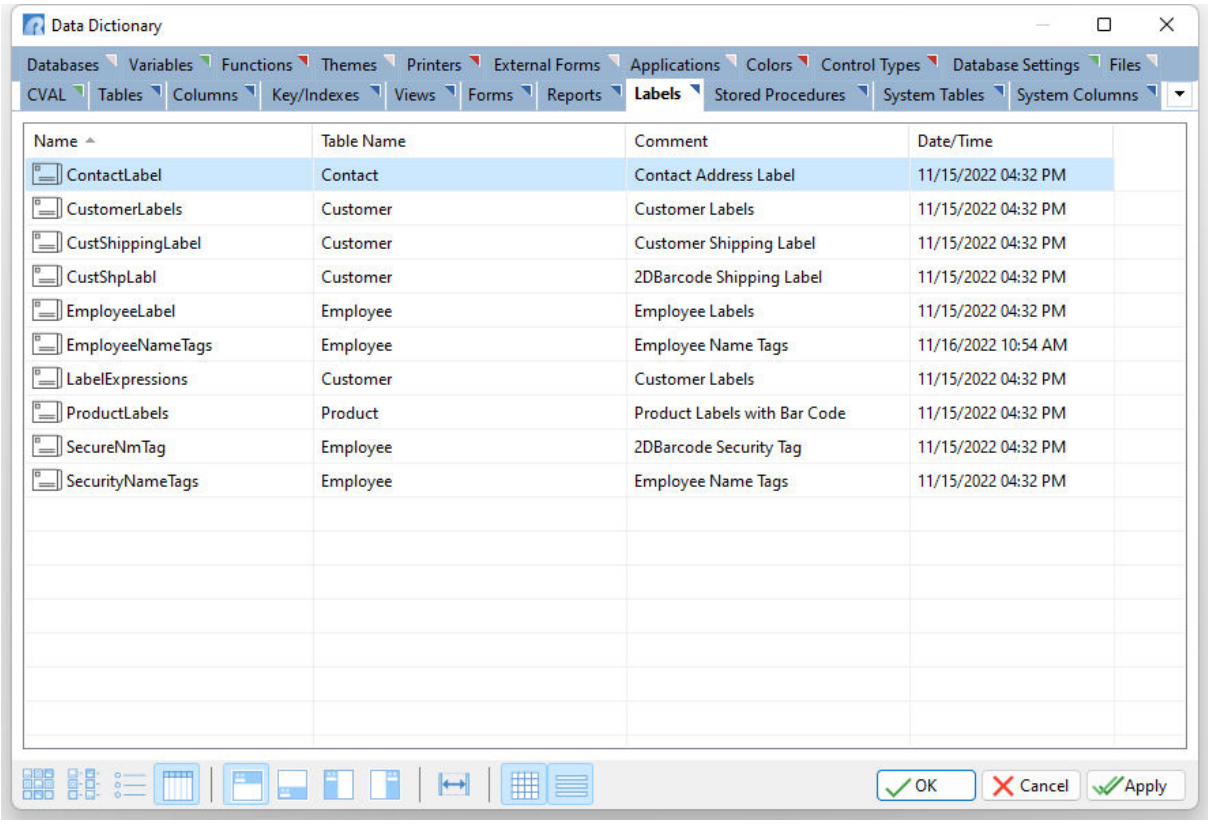
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.7 Labels

The Labels tab lists all defined labels in the current database. The label name, table name, comment, and last modified date/time is provided. With a label selected, pressing the OK button will capture the label name.

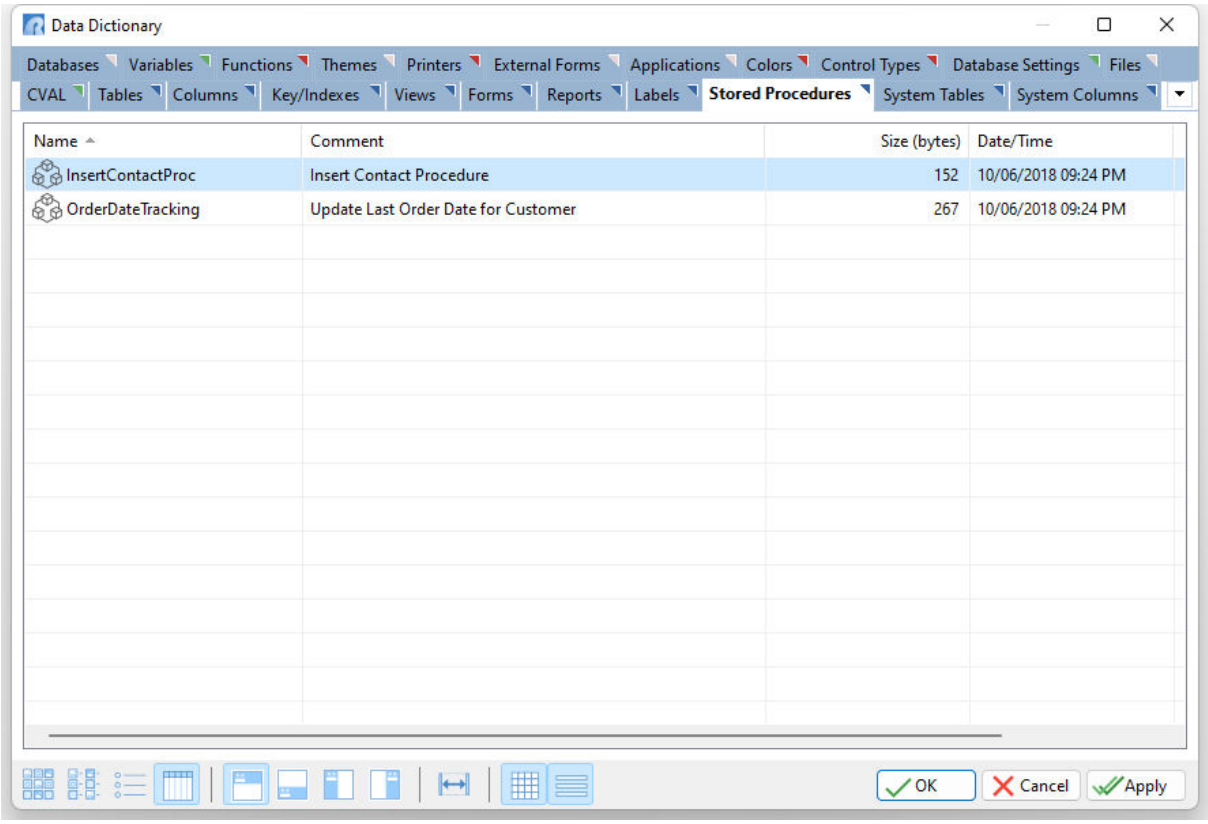
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.8 Stored Procedures

The Stored Procedures tab lists all defined stored procedures in the current database. The stored procedure name and comment is provided. With a stored procedure selected, pressing the OK button will capture the stored procedure name.

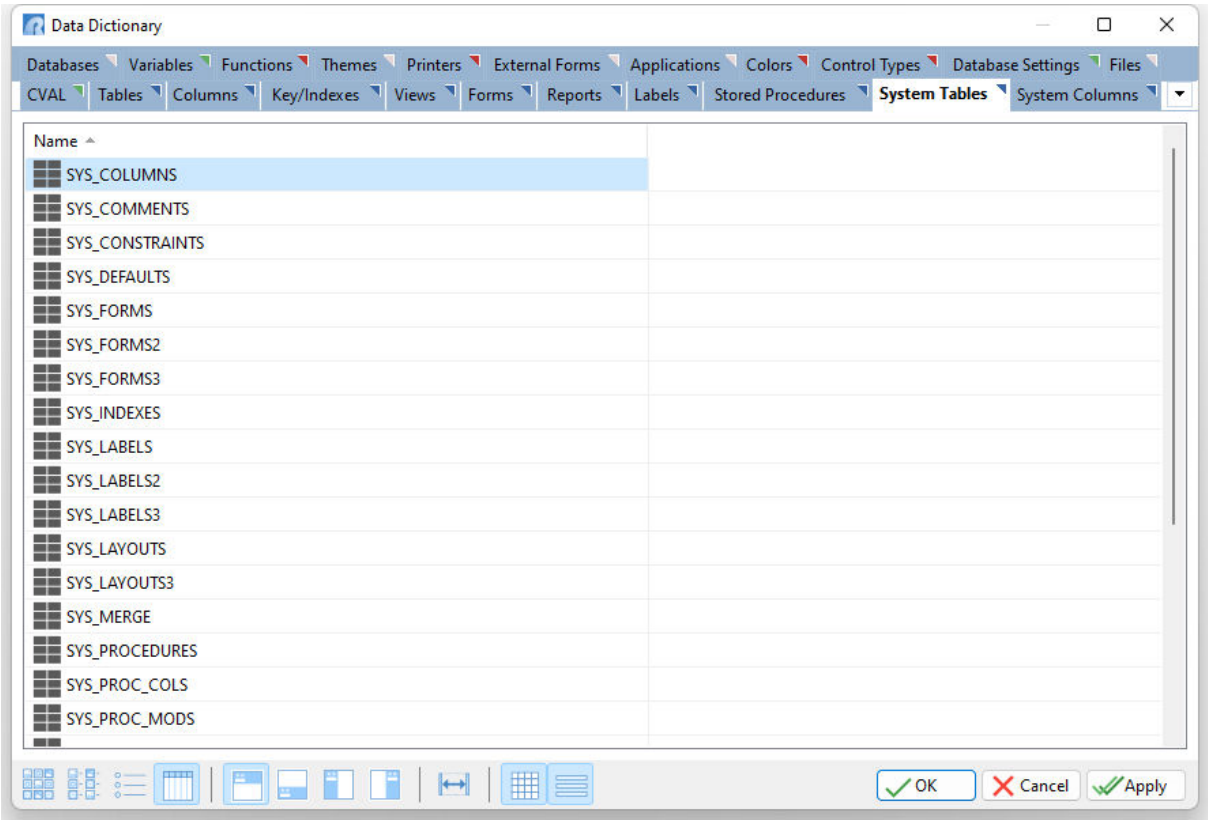
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.9 System Tables

The System Tables tab lists all R:BASE system tables. The system table name is provided. With a system table selected, pressing the OK button will capture the system table name.

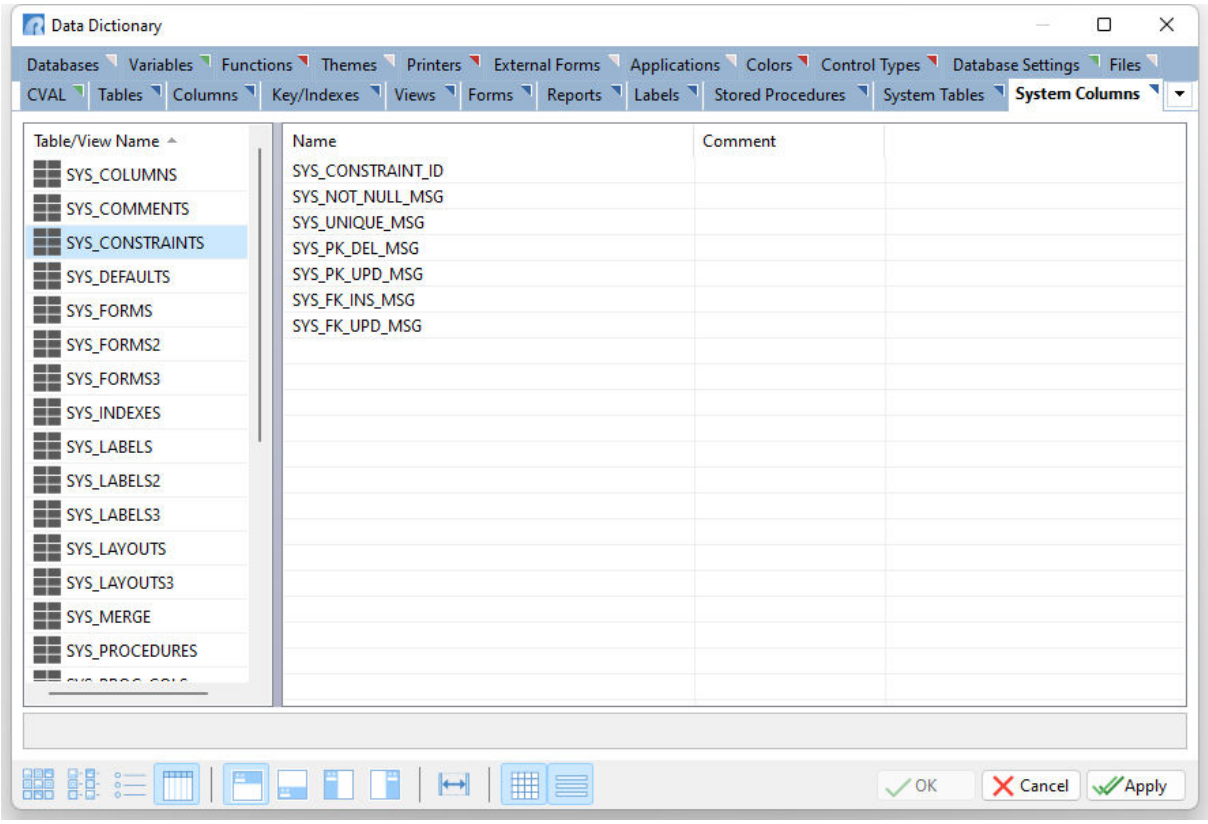
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.10 System Columns

The System Columns tab lists all system columns within the R:BASE system tables. The system column name and comment is provided. With a column selected the data type is provided. Pressing the OK button will capture the system column name. When selecting multiple system columns, the values captured are separated by a comma (or the current delimiter). The Space Separated button is available to add a space with the comma.

Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.11 Designer Tabs

The Data Dictionary will display additional tabs when the utility is launched within the Form Designer, Report Designer, and External Form Designer interfaces.

Within the Form Designer the Data Dictionary will display two additional tabs:

- Form Component IDs
- Form Actions

Within the Report Designer, the the Data Dictionary window will display one additional tab:

- Report Component IDs

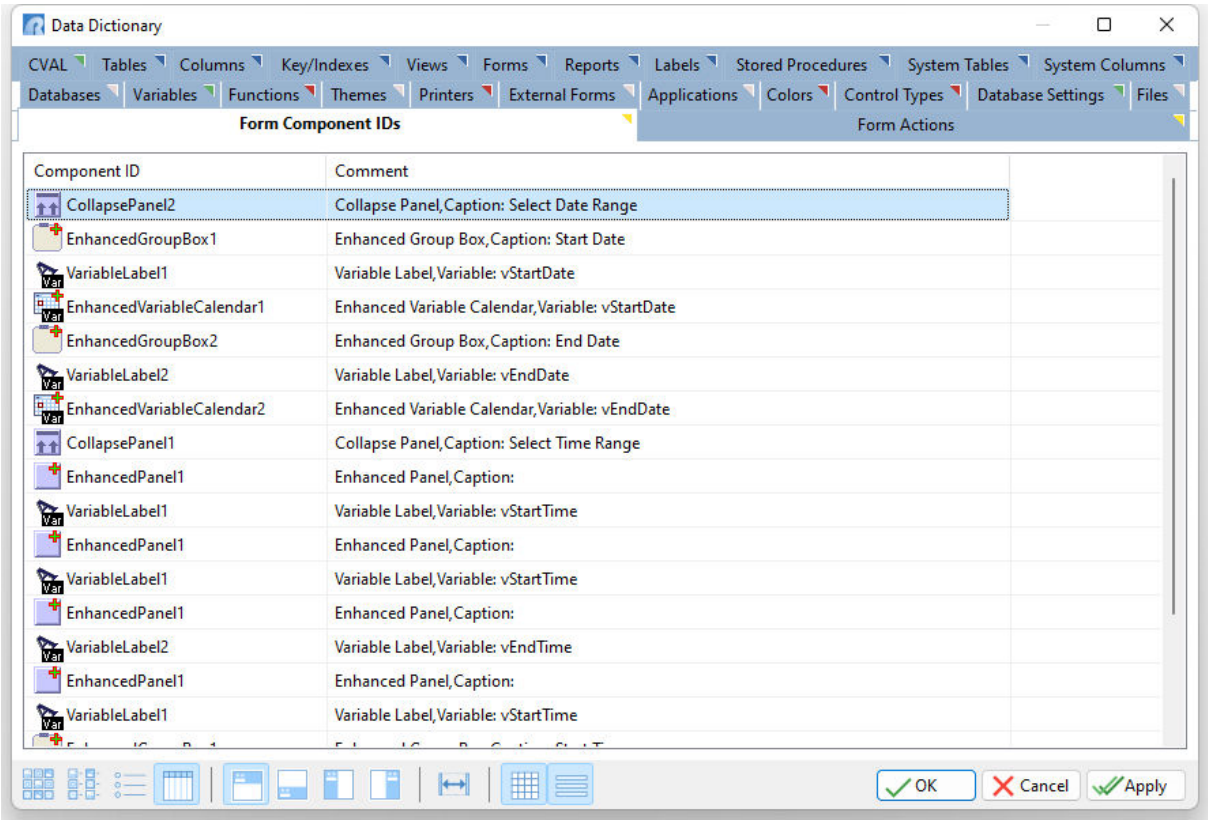
Within the External Form Designer the Data Dictionary will display two additional tabs:

- External Form Component IDs
- External Form Actions

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.42.2.11.1 Form Component IDs

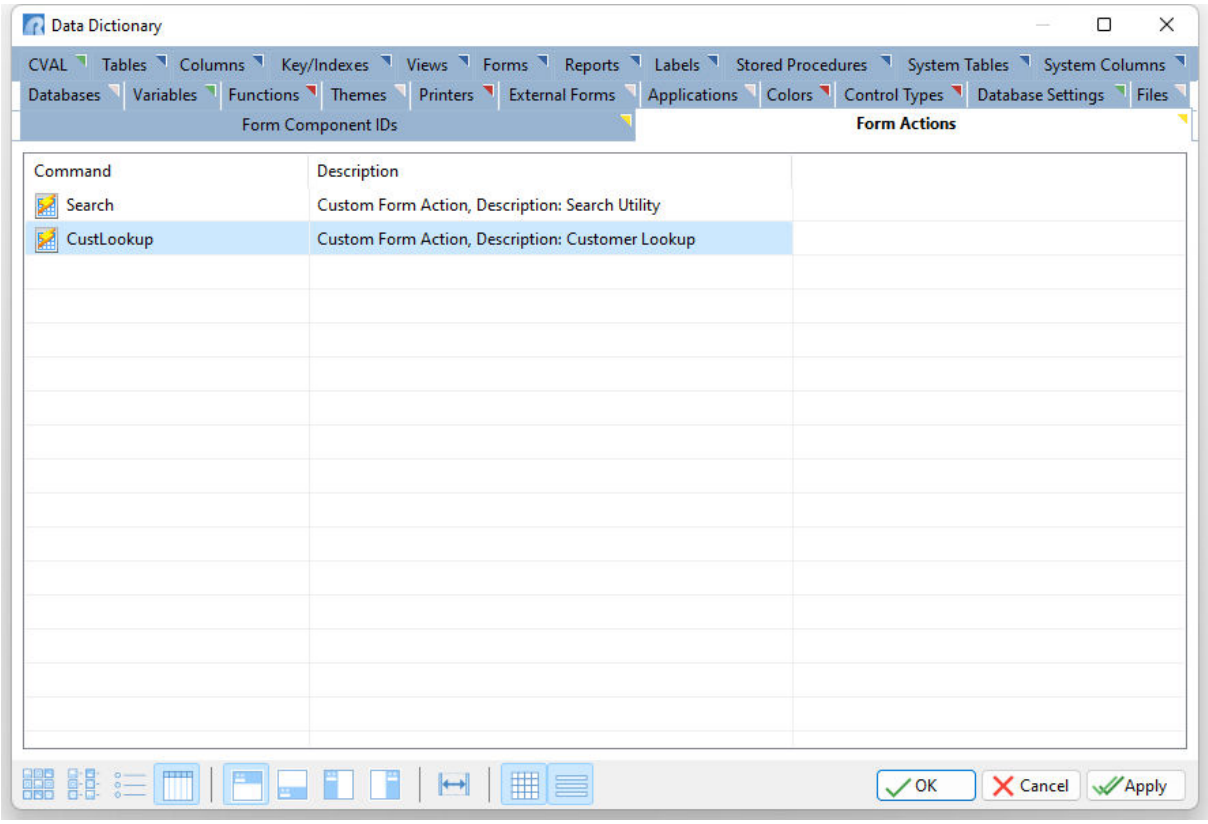
The Form Component IDs tab lists all Component IDs in the current form, when the Data Dictionary is launched within the Form Designer. The Component ID name and comment is provided. The comment will contain the control type, and caption (if defined), as well as other information based upon the control type. With a Component ID selected, pressing the OK button will capture the Component ID name.



1.42.2.11.2 Form Actions

The Form Actions tab lists all Custom Form Actions in the current form, when the Data Dictionary is launched within the Form Designer. The Custom Form Action command name and description is provided. With a Form Action selected, pressing the OK button will capture the Form Action command name.

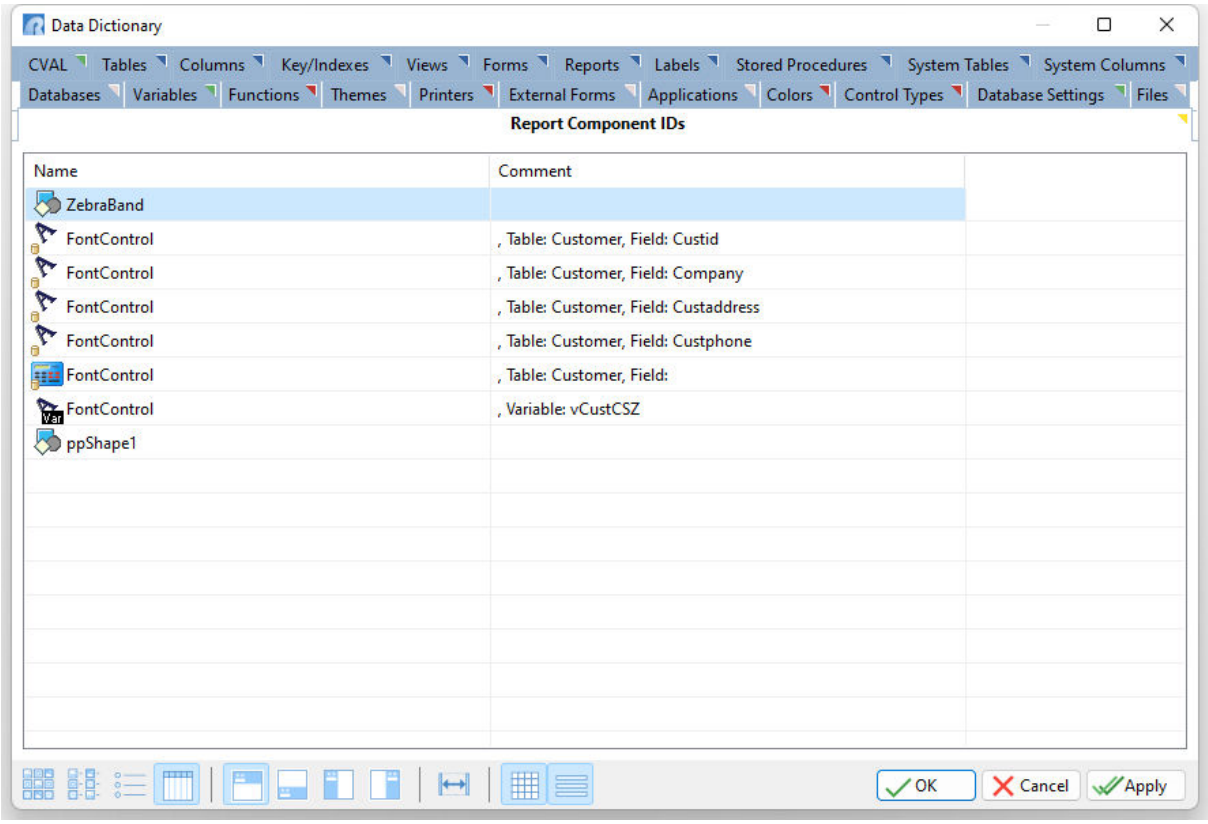
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.11.3 Report Component IDs

The Report Component IDs tab lists all Component IDs in the current report, when the Data Dictionary is launched within the Report Designer. The Component ID name and comment is provided. The comment will contain the control type, and caption (if defined), as well as other information based upon the control type. With a Component ID selected, pressing the OK button will capture the Component ID name.

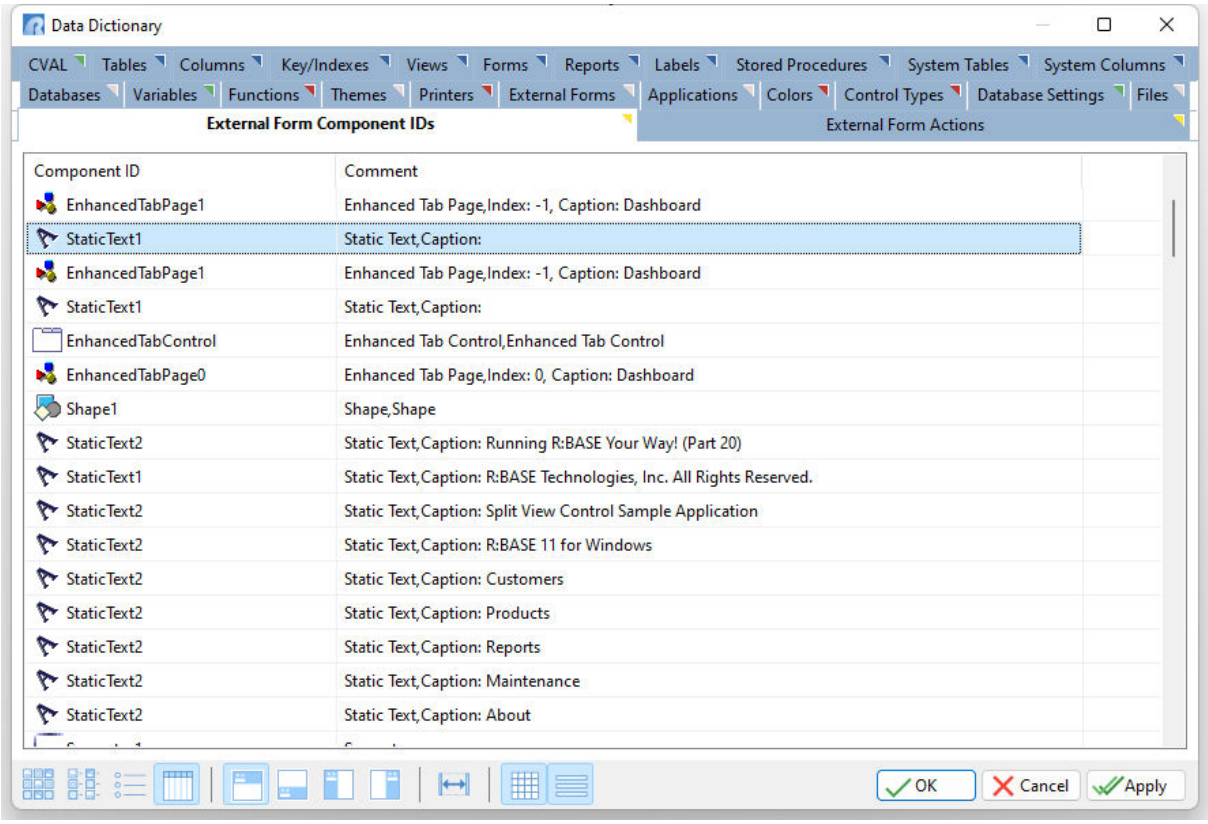
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.11.4 External Form Component IDs

The External Form Component IDs tab lists all Component IDs in the current external form file, when the Data Dictionary is launched within the External Form Designer. The Component ID name and comment is provided. The comment will contain the control type, and caption (if defined), as well as other information based upon the control type. With a Component ID selected, pressing the OK button will capture the Component ID name.

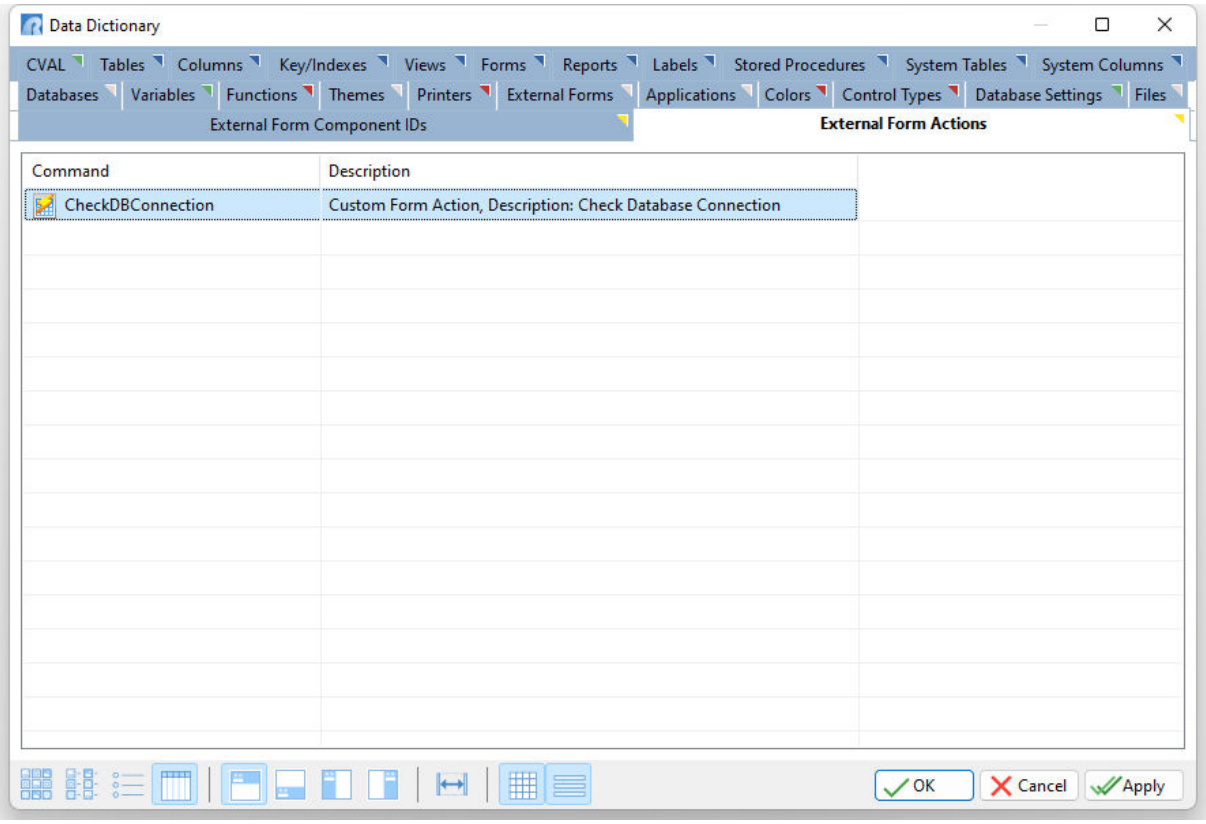
Copyright © 1982-2024 R:BASE Technologies, Inc.



1.42.2.11.5 External Form Actions

The External Form Actions tab lists all Custom Form Actions in the current external form file, when the Data Dictionary is launched within the External Form Designer. The Custom Form Action command name and description is provided. With a Form Action selected, pressing the OK button will capture the Form Action command name.

Copyright © 1982-2024 R:BASE Technologies, Inc.



1.43 Using the DLCALL Function

The DLCALL Function calls any Windows dynamic link library (DLL) and loads it into memory for use with R:BASE. A DLL is a Window's **D**ynamically **L**inked **L**ibrary.

Syntax for External DLL:

(DLCALL('libraryname.ext', 'FunctionOrProcedureName', [arg],[arg],[.....]))

Syntax for Windows API:

(DLCALL('libraryname', 'FunctionOrProcedureName', [arg],[arg],[.....]))

External DLLs must have the file name listed with the extension, such as 'MyLibrary.dll'.

Windows APIs require only the name of the Library, without the extension, such as: 'Kernel32' or 'User32'.

The DLLs must be created as Standard Windows 32-bit DLLs. Any number of functions or procedures can be used in a single DLL. Functions or Procedures to be used with DLCALL must be Exported in the DLL. No special code is necessary in the DLL for it to be used by R:BASE.

1.43.1 DLL Location

DLLs can be located in the Legal Windows Search Path, and If elsewhere, then specify the Full Path Name in DLLoad.

Search Path Used by Windows to Locate a DLL

Copyright © 1982-2024 R:BASE Technologies, Inc.

With both implicit and explicit linking, Windows first searches for "known DLLs", such as Kernel32.dll and User32.dll. Windows then searches for the DLLs in the following sequence:

1. The directory where the executable module for the current process is located.
2. The current directory.
3. The Windows system directory. The GetSystemDirectory function retrieves the path of this directory.
4. The Windows directory. The GetWindowsDirectory function retrieves the path of this directory.
5. The directories listed in the PATH environment variable.

1.43.2 When or If DLLOAD is Used

DLLOAD may be called "anytime" during the Session to Load the Library into Memory OR as a subsequent call to determine if the Library is Loaded.

In any event, DLLOAD is called internally when the Library is first referenced by DLCALL, and THEN the Library remains in memory until either the R:BASE session is ended OR DLFREE is called.

See also:

CHKFUNC
DELFUNC

1.43.3 Data Type Rules

Data Types in the functions must be of the Same Storage Size as the corresponding R:BASE Data Types.

Example: 32-bit Win32 Integer = 4 bytes of storage vs 32-bit R:BASE Integer = 4 bytes of storage

1.43.4 Declaration Logic

Calls to a function OR procedure from ANY DLL must be DECLARED at Least ONCE in the session in which they will be referenced.

Two Calling Conventions are supported, using the STDCALL and CDECL Keyword in the Declaration.

STDCALL

Function Declaration using STDCALL:

```
STDCALL FUNCTION 'functionName' (PTR VARCHAR (SIZE), ..... ) : VARCHAR
(SIZE)
STDCALL FUNCTION 'functionName' ALIAS 'functionAliasName' (PTR TEXT
(SIZE), ..... ) : TEXT (SIZE)
```

Procedure Declaration using STDCALL:

```
STDCALL VOID FunctionOrProcedureThatHasNoReturnValue (PTR TEXT (SIZE))
```

Windows API Declaration using STDCALL:

```
STDCALL FUNCTION 'GetCurrentDirectoryA' ALIAS 'GetCurrentDir' (PTR TEXT,
INTEGER ) : INTEGER
```

CDECL

Copyright © 1982-2024 R:BASE Technologies, Inc.

Function Declaration using CDECL:

```
CDECL FUNCTION 'functionName' (INTEGER) : INTEGER
CDECL FUNCTION 'functionName' ALIAS 'functionAliasName' (PTR DOUBLE) :
DOUBLE
```

Procedure Declaration using CDECL

```
CDECL VOID FunctionOrProcedureThatHasNoReturnValue (PTR TEXT (SIZE))
```

Important Notes:

- 'SIZE' applies to TEXT and VARCHAR Data Types.
- Parameters in the Declaration are in the REVERSE order from the Actual Function or Procedure in the DLL.
- Parameters can be 0 to *n* Parameters of any legal R:BASE Data type.
- Function Names ARE CASE SENSITIVE in the Declaration ONLY.
- The Case must match the casing used in the DLL.
- Case is INSENSITIVE when used in DLCALL.

1.43.5 Remarks

- For best results, TEXT and VARCHAR data should be passed with the PTR (Pointer) Attribute and ANY VARCHAR data type Larger than 32K as a Parameter, MUST be passed as PTR.
- VARCHAR data type as a Return Value restricted to 32K, but Any SIZE up to 256MB can be passed as a Pointer to the R:BASE Variable. Modification of the Data Passed as Pointer must be on the data pointed to.
- It is important that the SIZE parameter on TEXT and VARCHAR be Specified to avoid creating excess buffer space that has to be created from the Declaration.

1.43.6 Examples

If the Function is Declared like this:

```
STDCALL function 'somefunction' ( ptr varchar (nn)) : integer
```

Then *nn* <= 256Mb.

If the Function is Declared like this:

```
STDCALL function 'somefunction' ( ptr varchar ) : integer
```

Then because SIZE is omitted, the buffer for VARCHAR will have default SIZE = 256MB.
*** AVOID THIS UNLESS THAT IS THE ACTUAL SIZE OF THE DATA TO BE PASSED!**

If the Function is Declared like this:

```
STDCALL function 'somefunction' ( integer ) : varchar(nn)
```

Then because SIZE is Specified, nn bytes <= 32K will be returned.

If the Function is Declared like this:

```
STDCALL function 'somefunction' ( integer ) : varchar
```

Then because SIZE is omitted, the buffer for VARCHAR will have default SIZE = 32K.

When SIZE is Specified, the data passed as parameter or as return value, if Greater than the SIZE, will be truncated to SIZE.

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.43.6.1 Delphi

Example of a DLL created in Delphi exporting three functions:

```
// Begin Dll Code
library DemoLib;

uses
  SysUtils, Classes;

{$R *.res}

function MultInt (NumIN : Integer) : Integer; stdcall;
begin
  Result := (NumIN * 2);
end;

function MultDbl (NumDbl : Double) : Double; stdcall;
begin
  Result := (NumDbl * 2);
end;

procedure LCaseByREF(DataIN : PChar); stdcall;
begin
  ansiStrLower(DataIN);
end;

function LCaseByVAL (DataIN : PChar) : PChar; Stdcall;
begin
  Result := ansiStrLower(DataIN);
end;

Exports MultInt, LCaseByREF, LCaseByVAL;

begin

end.
// End Dll Code
```

1.43.6.2 R:BASE

Example Usage From Within R:BASE:

```
-- BEGIN Demo.rmd
-- Declare the functions to be used from the DLL
STDCALL function 'MultInt' ( Integer ) : Integer
STDCALL VOID 'LCaseByREF' (ptr TEXT (30))
STDCALL function 'LCaseByVAL (ptr TEXT (60)) : TEXT (60)

--Set somme variables for use
Set VAR vTEXT TEXT = 'RBASE TECHNOLOGIES'
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
SET VAR vINT INTEGER = 128
SET VAR v1 INTEGER = 0

-- OPTIONALLY CALL DLLOAD
SET VAR v1 = (DLLOAD('DemoLib.dll'))
IF v1 = 0 THEN
  PAUSE 2 USING 'DemoLib.dll NOT LOADED.. EXITING'
  RETURN
ENDIF

SET VAR V1 = (dlcall('demolib.dll', 'changecase', vtext))

{ The Value for v1 will be null because ChangeCase is a procedure and
doesn't
  RETURN A RESULT, but the value of vTEXT which is passed as a POINTER has
been
  changed to 'rbase technologies'}

SET VAR v1 = (DLCALL('demolib.dll', 'MultInt', vINT))

--The value for v1 will be 256 the value returned from the function.

-- running the following against RRBYW14
SELECT (DLCALL('demolib.dll','lcasebyval', Company))=60 FROM +
Customer WHERE LIMIT = 2

{Yields the following output:
 (DLCALL('demolib.dll','lcasebyval', Company)
 ------------------------------------------------------------
 computer warehouse - ii
 microtech university - i
}

SELECT ((ICAP2((DLCALL('demolib.dll','lcasebyval', Company))))) = 60 +
FROM Customer WHERE LIMIT = 2

{Yields the following otput:
 ((ICAP (DLCALL('demolib.dll','lcasebyval',
 ------------------------------------------------------------
 Computer Warehouse - Ii
 Microtech University - I
}

-- Optionally CALL DLFREE
SET VAR v1 = (DLFREE('DemoLib.dll'))

-- END Demo.rmd
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

1.43.6.3 C++

Example of a DLL created in C++ Exporting three functions:

```
// BEGIN C++ DLL
// loaddll.cpp : Defines the entry point for the DLL application.
//
#include <windows.h>
#include <stdio.h>

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                                          )
{
    return TRUE;
}
#ifdef __cplusplus    // If used by C++ code,
extern "C" {          // we need to export the C interface
#endif

__declspec(dllexport) int cfunc1(int i){
        return i;
}
__declspec(dllexport) double cfunc2(double *inp){
        double rtn = *inp;
        rtn++;
        return rtn;
}
__declspec(dllexport) char * cfunc3(char *inp){
        strcat(inp," + ");
        return inp;
}

#ifdef __cplusplus
}
#endif

// END C++ DLL
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

Part II

2 Settings

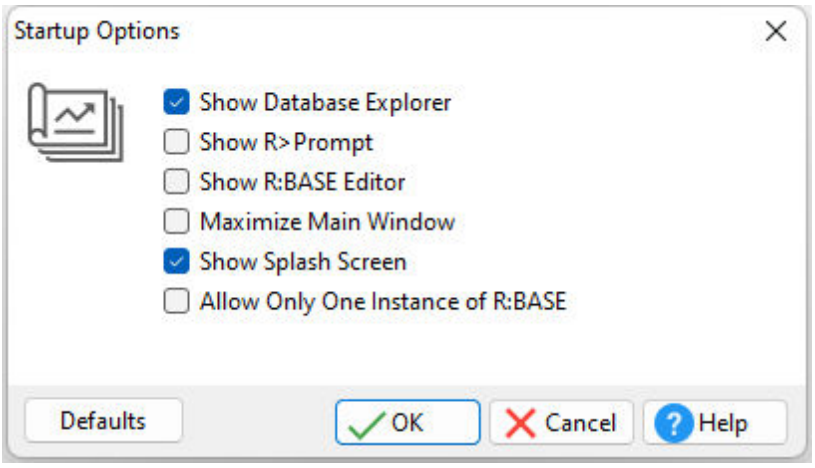
The R:BASE Settings allow you to alter various aspects of the R:BASE program. The R:BASE Settings are accessible from the main Menu Bar under "Settings", and are available at any time in the development environment.

2.1 Startup Options

The Startup Options selection allows you to customize how R:BASE initially starts.

- *Show Database Explorer* - will display the Database Explorer
- *Show R> Prompt* - will display the R> Prompt
- *Show R:BASE Editor* - will display the R:BASE Editor
- *Maximize Main Window* - will maximize the R:BASE window
- *Show Splash Screen* - will display the R:BASE splash screen
- *Allow Only One Instance of R:BASE* - limits the number of R:BASE instances that can be launched to one

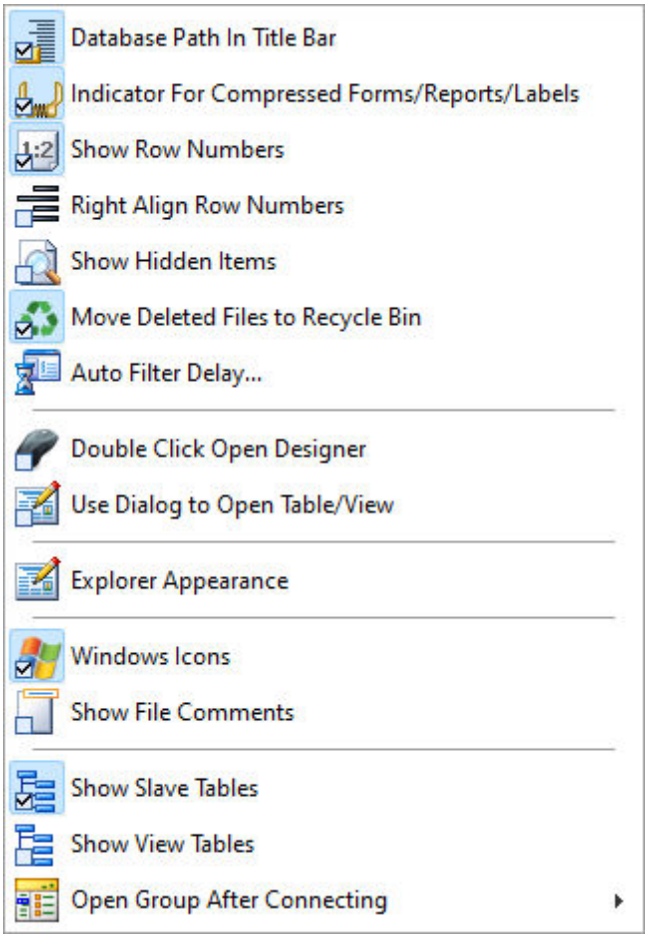
- *Defaults* - Restores the default settings for the startup options



Copyright © 1982-2024 R:BASE Technologies, Inc.

2.2 Database Explorer

The Database Explorer selection allows you to change the appearance of the Database Explorer window.



- *Database Path in Title Bar* - toggles the display of the path for the connected database
- *Indicator For Compressed Forms/Reports/Labels* - toggles the display of the "vice" indicator for compressed objects
- *Show Rows Numbers* - toggles the display of row numbers for objects
- *Right Align Row Numbers* - adjusts the row number alignment
- *Show Hidden Items* - toggles the display of files, based upon the "hidden" file attribute. Hidden files may include databases, external form files, applications, and command files.
- *Move Deleted Files to Recycle Bin* - toggles sending deleted files to the Recycle Bin or not. The default is enabled. File-based objects (external form files (.rff), application files (.rba), command files, and database files (.RX1-.RX4) are sent to the Recycle Bin.
- *Auto Filter Delay...* - specifies the interval for how often keystrokes are recognized within the "Find in list" filter [Alt+Q]
- *Double Click Open Designer* - toggles the double click "run" or "design" behavior for items displayed
- *Use Dialog to Open Table/View* - toggles the ability to launch a dialog to add a WHERE Clause, change the browse mode, and use an MDI window when opening a table/view. The dialog also displays the available WHERE Clause history [F5].
- *Explorer Appearance* - adjusts the color schema of R:BASE and the Database Explorer areas
- *Windows Icons* - uses the Windows icons for files listed under "Command Files" within the Group Bar
- *Show File Comments* - displays the "File Comment" column in the Commands Files area. Comments for command files will be interpreted where braces {...} are used in the first line of the file.
- *Show Slave Tables* - toggles the display of slave tables for forms, reports, and labels
- *Show View Tables* - toggles the display of tables/views for views.

Copyright © 1982-2024 R:BASE Technologies, Inc.

When enabled, the setting can slow down response times when many/complicated views are present.
- *Open Group After Connecting* - sets a default group to be displayed after connecting to a database

2.3 R:BASE Editor...

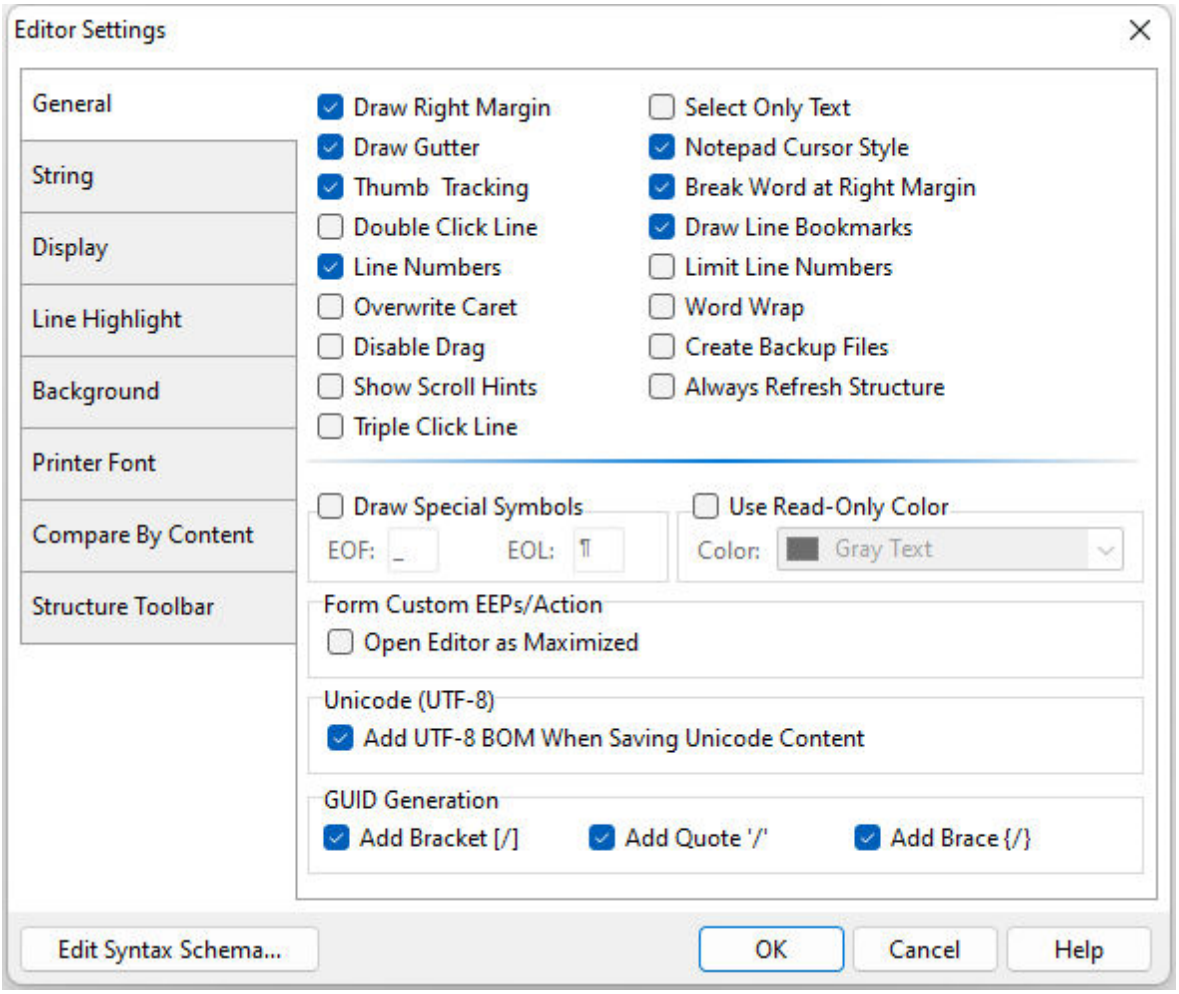
This menu presents the configuration panel for the R:BASE Editor settings.

2.3.1 General

- *Draw Right Margin* - enables/disables a right margin in the command file
- *Draw Gutter* - enables/disables a gutter on the left margin
- *Thumb Tracking* - enables/disables the command file movement before the thumb tab is released while sliding the scroll bar
- *Double Click Line* - highlights the line when you double-click any character in the line
- *Line Numbers* - enables/disables line numbering in the gutter
- *Overwrite Caret* - enables/disables overwrite cursor when the editor is in overwrite mode
- *Disable Drag* - disables drag and drop of selected text
- *Show Scroll Hints* - displays the line number when dragging the vertical scroll bar
- *Triple Line Click* - highlights the line when you triple-click any character in the line
- *Select Only Text* - only text is selected when additional lines selected
- *Notepad Cursor Style* - allows cursor to not stay on first character in line when left arrow is pressed
- *Break Word at Right Margin* - enables/disables word break at the right margin when word wrap is on
- *Draw Line Bookmarks* - enables/disables an arrow placement at the cursor location of an added bookmark
- *Limit Line Numbers* - enables/disables limitation line numbers to the length of a command file
- *Word Wrap* - enables/disables wrapping text
- *Create Backup Files* - enables/disables the creation of a backup file once a file is saved (file = CUST.APP backup = CUST.~AP , file = RESTORE.RMD backup = RESTORE.~RM)
- *Always Refresh Structure* - refreshes the Structure tool bar when switching between open files and when the file is saved
- *Draw Special Symbols* - allows the display of special symbols for the end of a file (EOF) and the end of a line (EOL)
- *Use Read-Only Color* - allows the display of a color background for read-only lines
- *Form Custom EEPs/Action* - enables/disables the editor window to be maximized when using Custom EEPs and Actions
- *Unicode (UTF-8)* - specifies whether to add a UTF-8 BOM to a saved file

Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor

Copyright © 1982-2024 R:BASE Technologies, Inc.



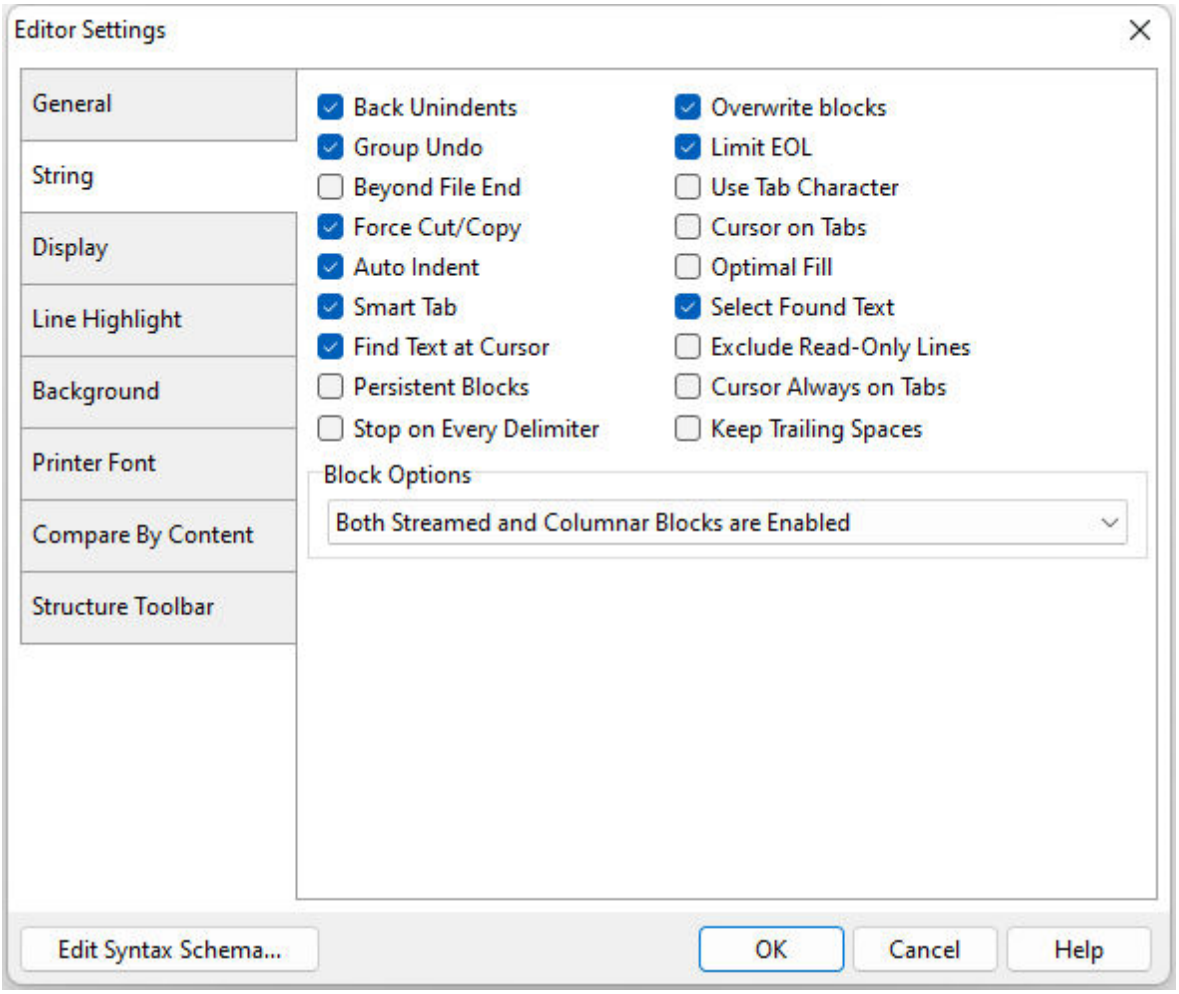
2.3.2 String

- *Back Unindents* - enables/disables alignment of the cursor with the previous indentation level from the preceding line when you press the BACKSPACE key
- *Group Undo* - enables/disables the undoes of your last editing command as well as any subsequent editing commands of the same type
- *Beyond File End* - enables/disables position of the cursor beyond the last character
- *Force Cut/Copy* - enables/disables cut and copy, even when there is no text selected
- *Auto Indent* - enables/disables the cursor to be placed under the first non-white space character in the preceding line when you press the ENTER key
- *Smart Tab* - enables/disables the cursor to be placed under the first non-white space character in the preceding line when you press the TAB key
- *Find Text at Cursor* - enables/disables placement of the text at the cursor into the Text To Find list box in the Find Text dialog box
- *Persistent Blocks* - enables/disables to keep a marked block (selected text) selected even when the cursor is moved
- *Stops on Every Delimiter* - Enables/disables cursor navigation mimicking the R> cursor, where pressing [Ctrl] + [Arrow] Keys will move the cursor to the far left character for every word or combined character string.
- *Overwrite Blocks* - enables/disables to replace a marked block (selected text) with whatever is typed next
- *Limit EOL* - enables/disables the cursor from being positioned past the end of a line
- *Use Tab Character* - enables/disables insert of the Tab character into the text when the user presses Tab button

Copyright © 1982-2024 R:BASE Technologies, Inc.

- *Cursor on Tabs* - enables/disables the caret to be moved by tabs
- *Optional Fill* - enables/disables the beginning of every auto-indented line with the minimum number of characters possible, using tabs and spaces as necessary
- *Select Found Text* - enables/disables to select the text that was found in a Find operation
- *Exclude Read-Only Lines* - enables/disables excluding read-only text
- *Cursor Always on Tabs* - enables/disables pressing right and left arrows will move the cursor on tabs
- *Keep Trailing Spaces* - keeps/removes trailing spaces at the end of code lines
- *Block Options* - options for block selection in command files

Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor



2.3.3 Display

⇨ **Gutter**

- *Width (Screen Pixels)* - options for width of gutter (0 will base the gutter width on file size)
- *Color* - options for the gutter color
- *Line Number Color* - options for color of numbers in the gutter

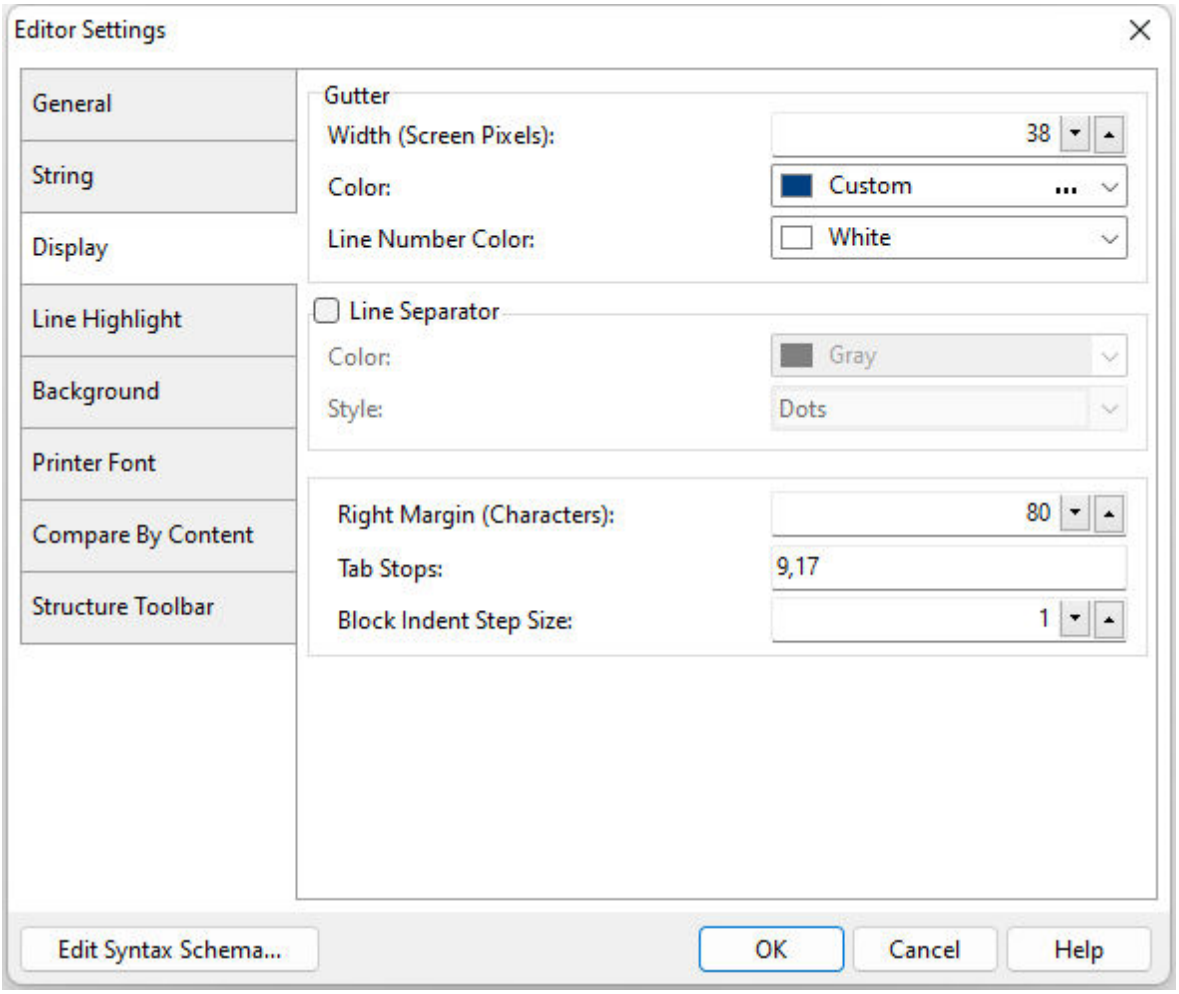
- *Line Separator* - enables/disables a separator between number lines of code with color and style

- *Right Margin (Characters)* - options for placement of right margin
- *Tab Stops* - stop points for tabbing across a line

Copyright © 1982-2024 R:BASE Technologies, Inc.

- *Block Indent Step Size* - number of spaces that a command block will indent using [Ctrl]+[K]+[I]; outdent using [Ctrl]+[K]+[U]

Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor



2.3.4 Line Highlight

These settings will "highlight" the current selected line with a style, color, and background where the cursor is located.

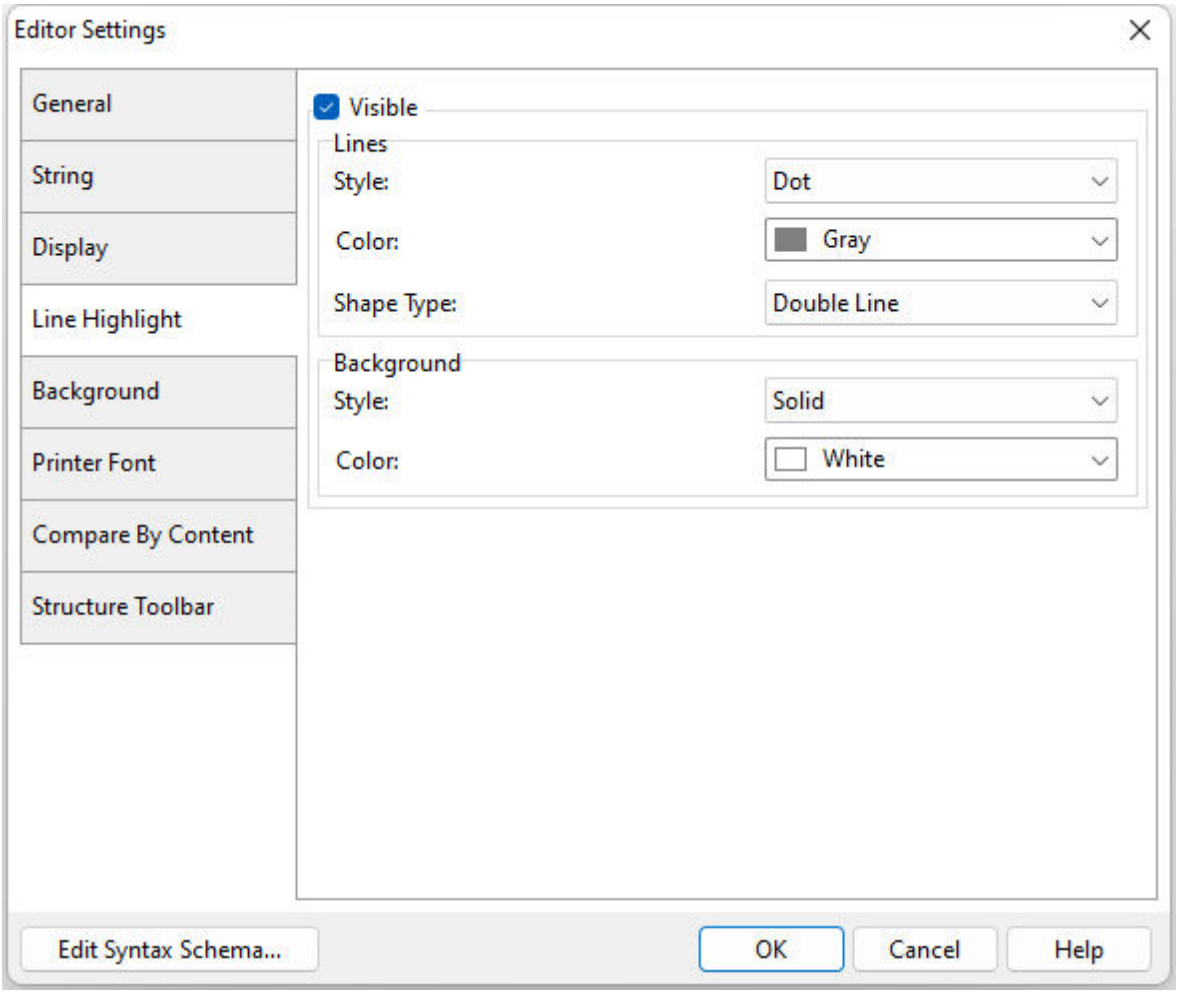
Visible - enables/disables line highlighting

⇨ **Lines**
Style - specifies the line style
Color - specifies the color of the line(s)
Shape Type - specifies the number of lines to display

⇨ **Background**
Style - specifies the background style
Color - specifies the background color

Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.3.5 Background

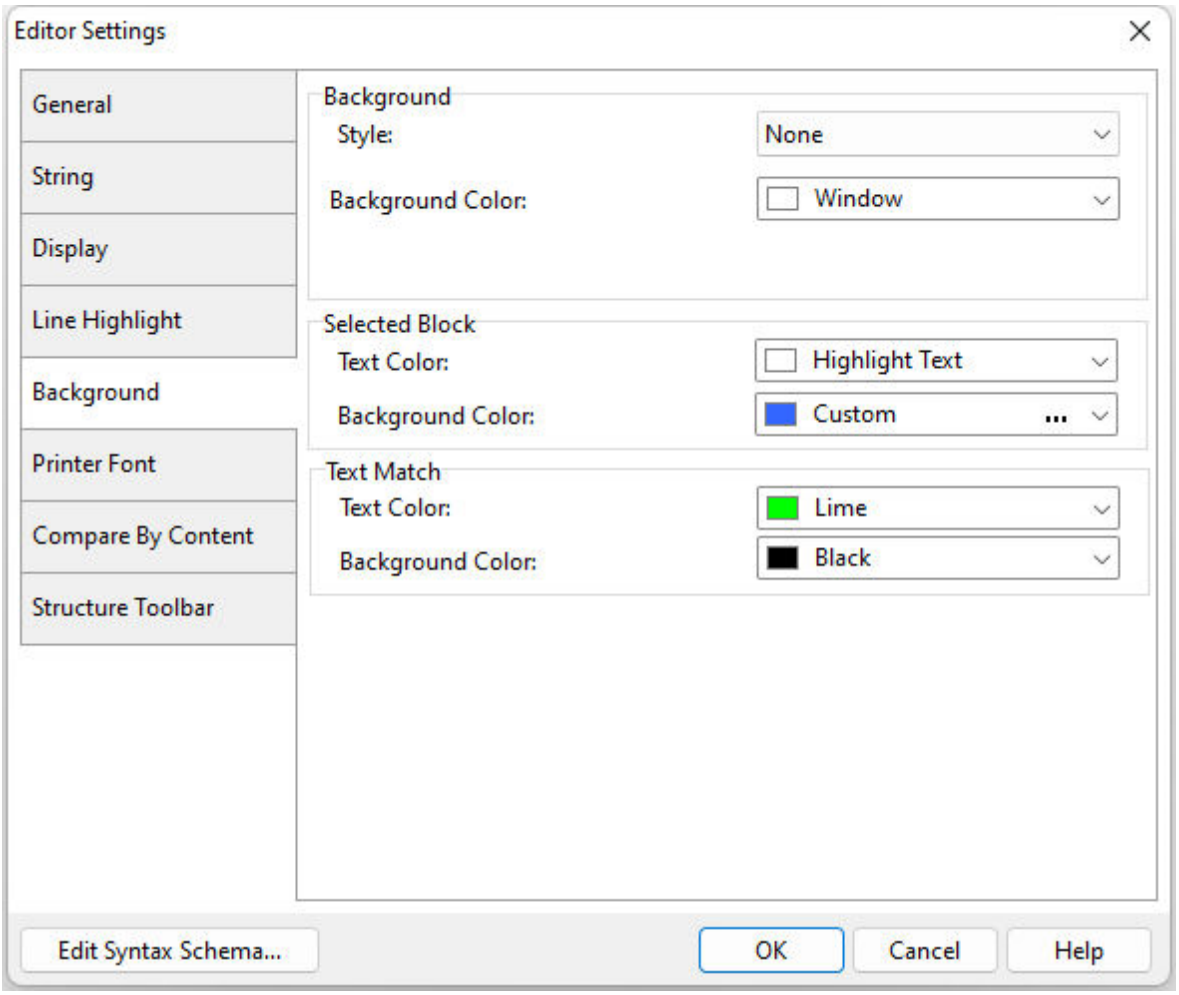
⇨ **Background**
Will alter the background of the applied to the Editor window, the File Structure toolbar, and the Error Messages toolbar
Style - specifies a background style supporting an image or gradient
Background color - specifies the color of the background

⇨ **Selected Block**
Text Color - specifies the font color of selected text
Background color - specifies the background color of selected text

⇨ **Text Match**
Text Color - specifies the font color of matching text when using the Find Text utility
Background color - specifies the background color of matching text when using the Find Text utility

Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor

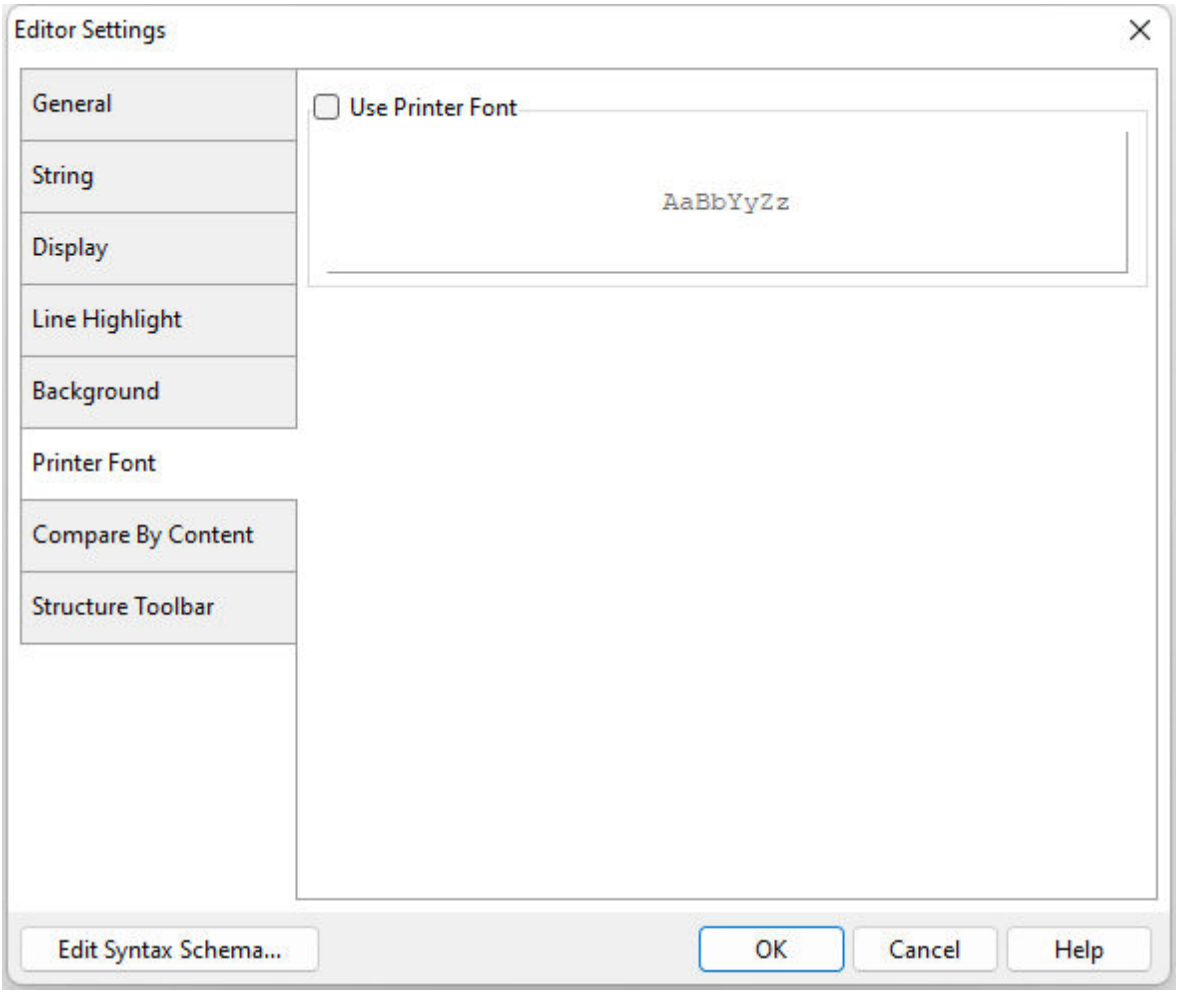
Copyright © 1982-2024 R:BASE Technologies, Inc.



Copyright © 1982-2024 R:BASE Technologies, Inc.

2.3.6 Printer Font

Printer output can be forced to print any of the available fonts.



Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor

2.3.7 Compare by Content

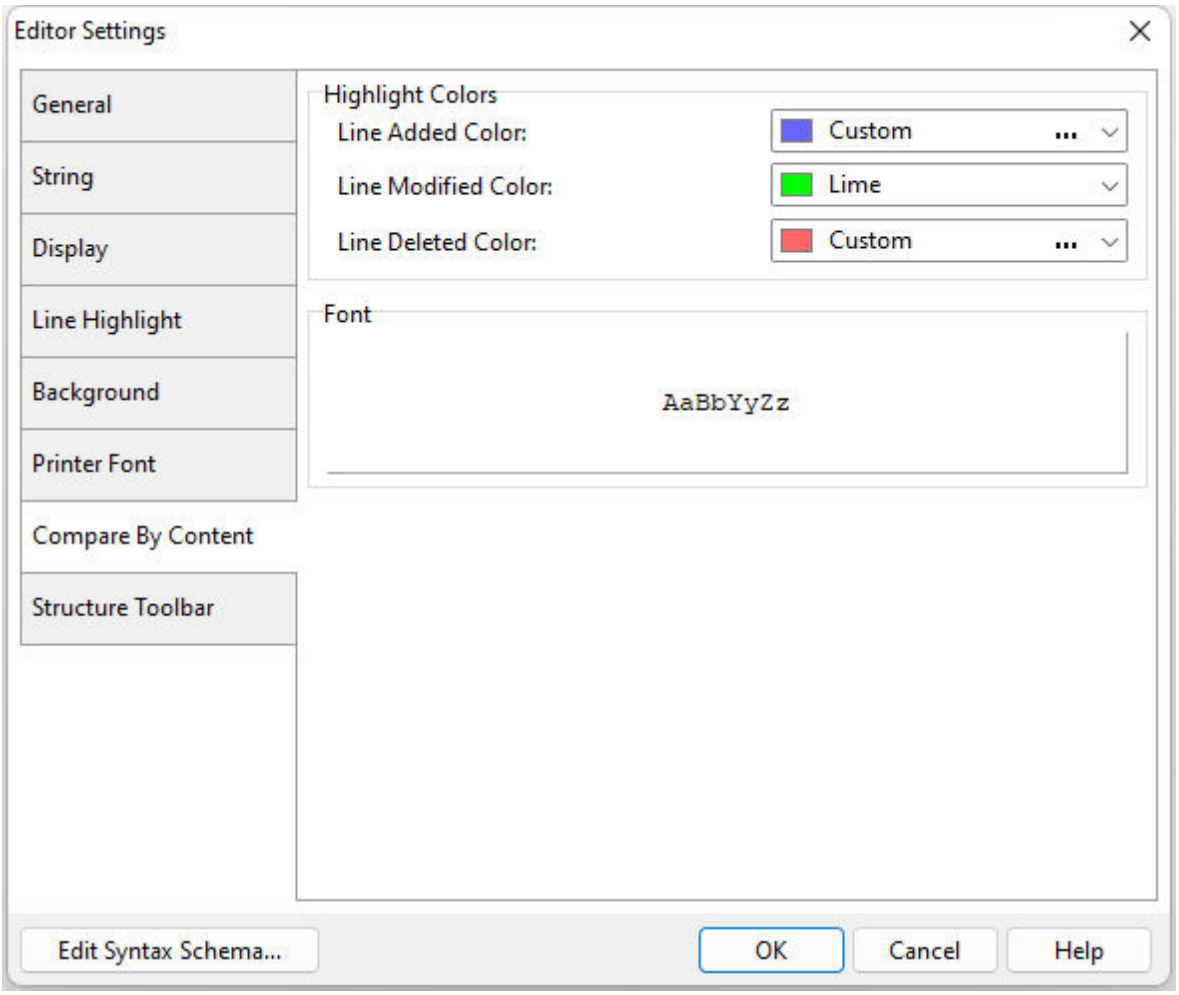
The colors which are displayed when comparing two commands file can be changed to any of the available colors. The font displayed in the Compare by Content Toolbar can also be changed.

⇨ **Highlight Colors**
Line Added Color - specifies the line color of text that has been added to the file
Line Modified Color - specifies the line color of text that has been modified within the file
Line Deleted Color - specifies the line color of text that have been deleted from the file

⇨ **Font**
Specifies the font displays in the Compare by Content Toolbar

Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor

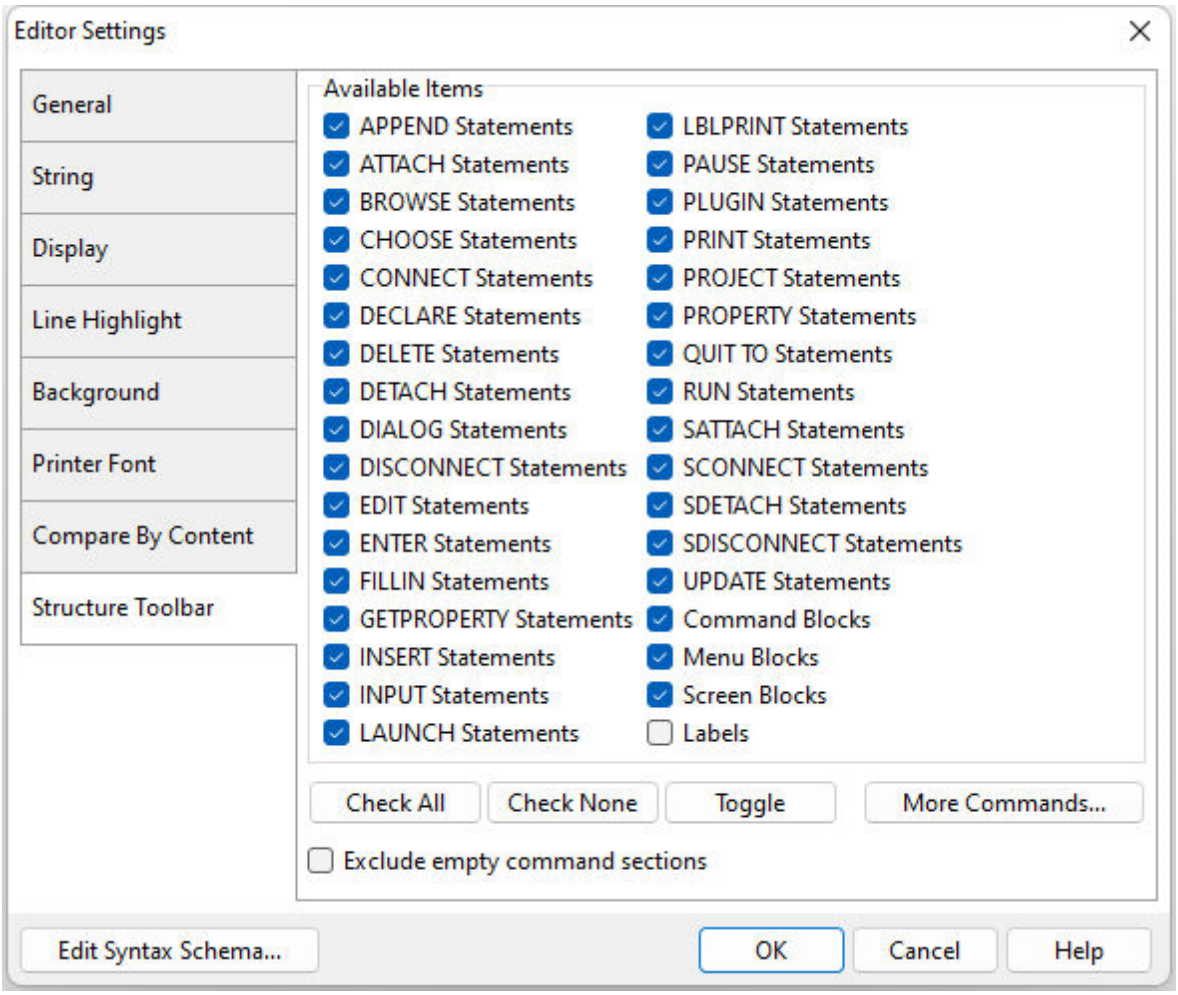
Copyright © 1982-2024 R:BASE Technologies, Inc.



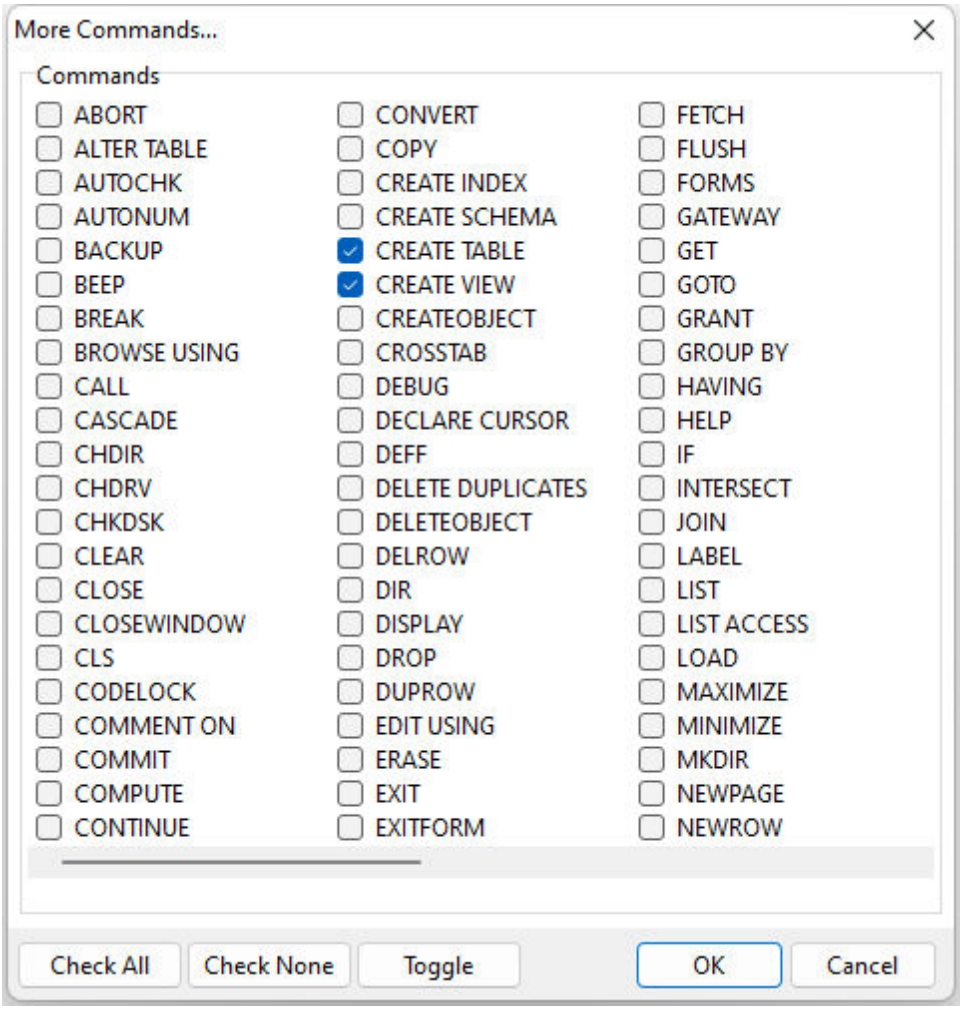
2.3.8 Structure Toolbar

The Structure Toolbar displays command file blocks and specific statements. Items can be removed from being listed in the toolbar by unselecting the available check boxes.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Copyright © 1982-2024 R:BASE Technologies, Inc.



Exclude empty command sections in Output Structure - excludes commands from the output when not listed in the command file
Edit Syntax Schema... - displays the Schema Editor to alter the syntax highlighting that affects the way code is displayed in the editor

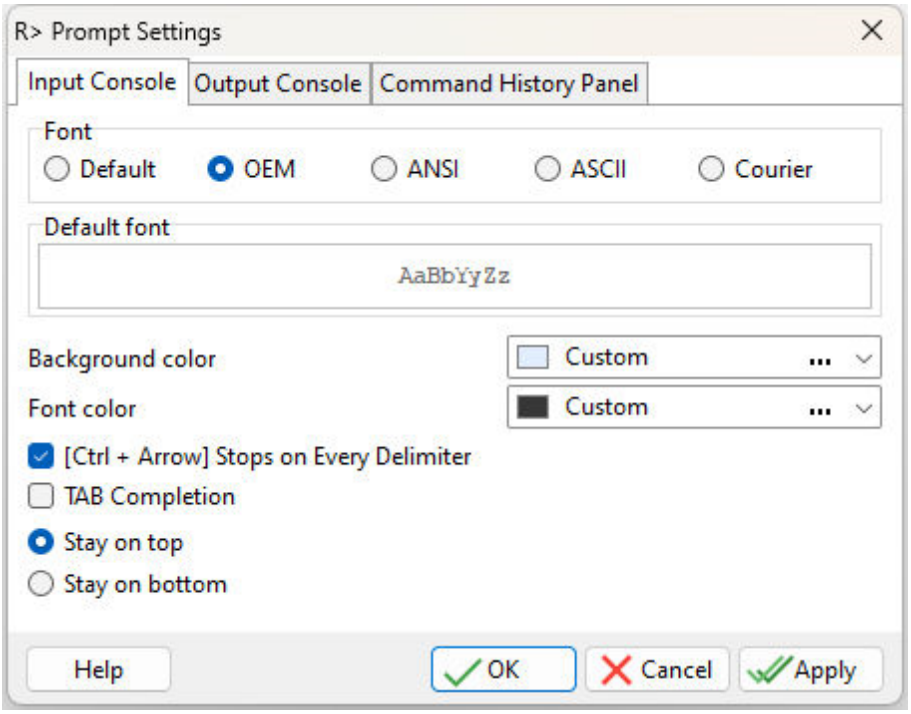
2.4 R> Prompt...

The R> Prompt Settings allows you to alter the font style and color, background colors, and other settings within the R> Prompt window. To access the R> Prompt Settings, select "Settings" > "R> Prompt" from the main Menu Bar and the following window will appear:

⇨ **Input console**
Determines the settings for the input portion of the R> Prompt console
Font
Specifies the font for the input console. To use a font other than OEM, ANSI, ASCII, or Courier, enable the "Default" option, and use the "Default font" area to assign the desired font properties.
Default font
Specifies the font style, size, and color for the input console, when the "Default" option is enabled.
Background Color
Specifies the background color of the input console
Font Color
Specifies the font color for the input console
[Ctrl + Arrow] Stops on Every Delimiter
When checked, the R> Prompt cursor will stop on every delimiter value

Copyright © 1982-2024 R:BASE Technologies, Inc.

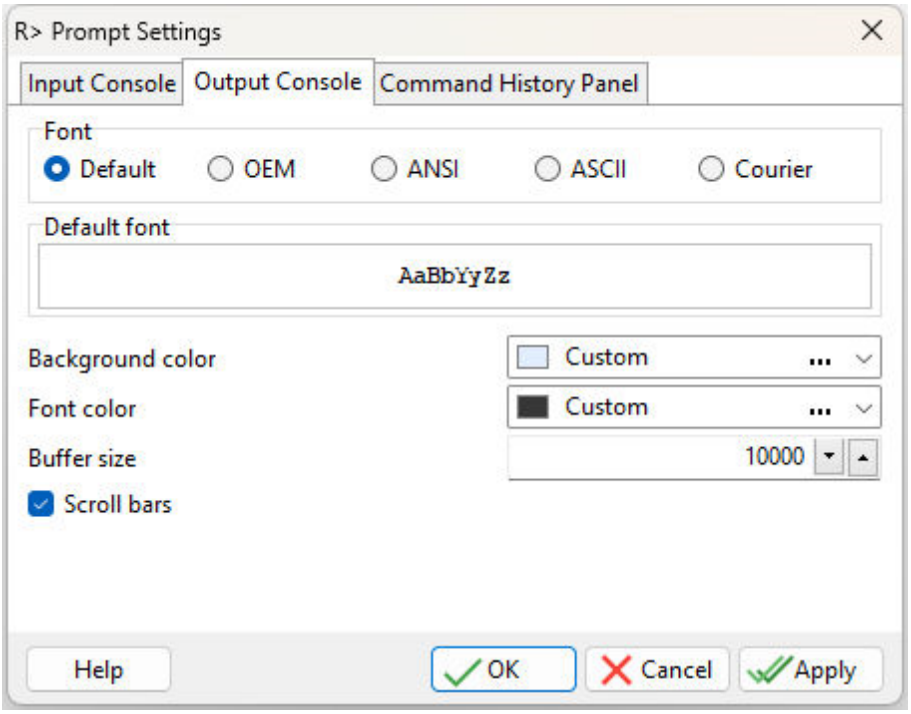
TAB Completion
When enabled, the input console will complete the syntax for a command when the [Tab] key is pressed
Stay on Top
Keeps the input console position above the output console
Stay on Bottom
Keeps the input console position below the output console



⇨ **Output console**
Determines the settings for the output portion of the R> Prompt console

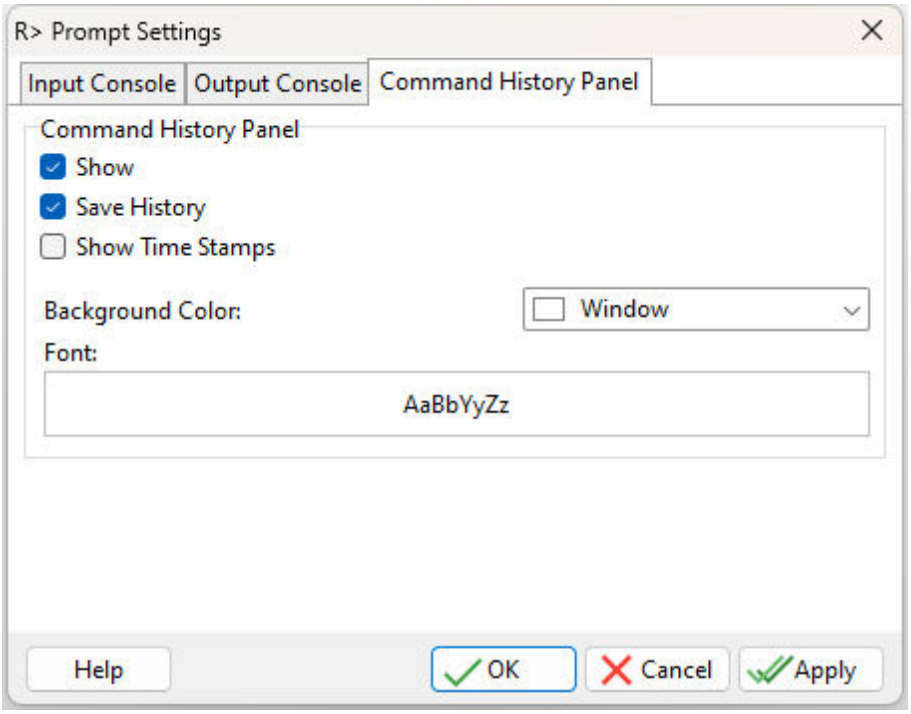
Font
Specifies the font for the output console. To use a font other than OEM, ANSI, ASCII, or Courier, enable the "Default" option, and use the "Default font" area to assign the desired font properties.
Default font
Specifies the font style, size, and color for the output console, when the "Default" option is enabled.
Background Color
Specifies the background color of the output console
Font Color
Specifies the font color for the output console
Buffer Size
Determines the amount of output which will be made available in memory
Scroll Bars
When checked, scroll bars will appear to view output which is out of the screen range

Copyright © 1982-2024 R:BASE Technologies, Inc.



⇨ **Command History Panel**

Show - specifies whether the "Command History" panel is displayed
Save History - specifies whether the "Command History" is loaded back into the panel after restarting R:BASE
Show Time Stamps - specifies to display the date/time stamp in the "Command History" to reference when a command was executed
Background Color - specifies the background color of the history panel
Font - specifies the font style, size and color for the history panel



Copyright © 1982-2024 R:BASE Technologies, Inc.

2.5 Data Browser...

The Data Browser settings allow you to change the display of the Data Browser/Editor window.

2.5.1 General Options

⇨ **Default View**
Specifies the default view format (Grid, Row, or Tree Grid)

⇨ **Background Color and Font**
Color - specifies the background color of the Data Browser/Editor
AaBbYyZz - specifies the font style, size and color of the Data Browser/Editor

⇨ **Draw Column Titles with Gradient**
Adds a color gradient to the table column titles
Start Color - specifies the starting gradient color
End Color - specifies the ending gradient color

⇨ **Show Row Number**
Displays the record number in a column to the far left

⇨ **Open Maximized**
Will automatically maximize the Data Browser

⇨ **Column Lines**
Displays the vertical column lines

⇨ **Row Lines**
Displays the horizontal row lines

⇨ **Grid Focus Navigation**
Specifies whether the cursor navigation on fields require the [Enter] key to manipulate a cell contents and move up and down. When enabled, the [Enter] key is required in order to exit a field and move from row to row after editing.

⇨ **Multiline Titles**
Supports multiline titles

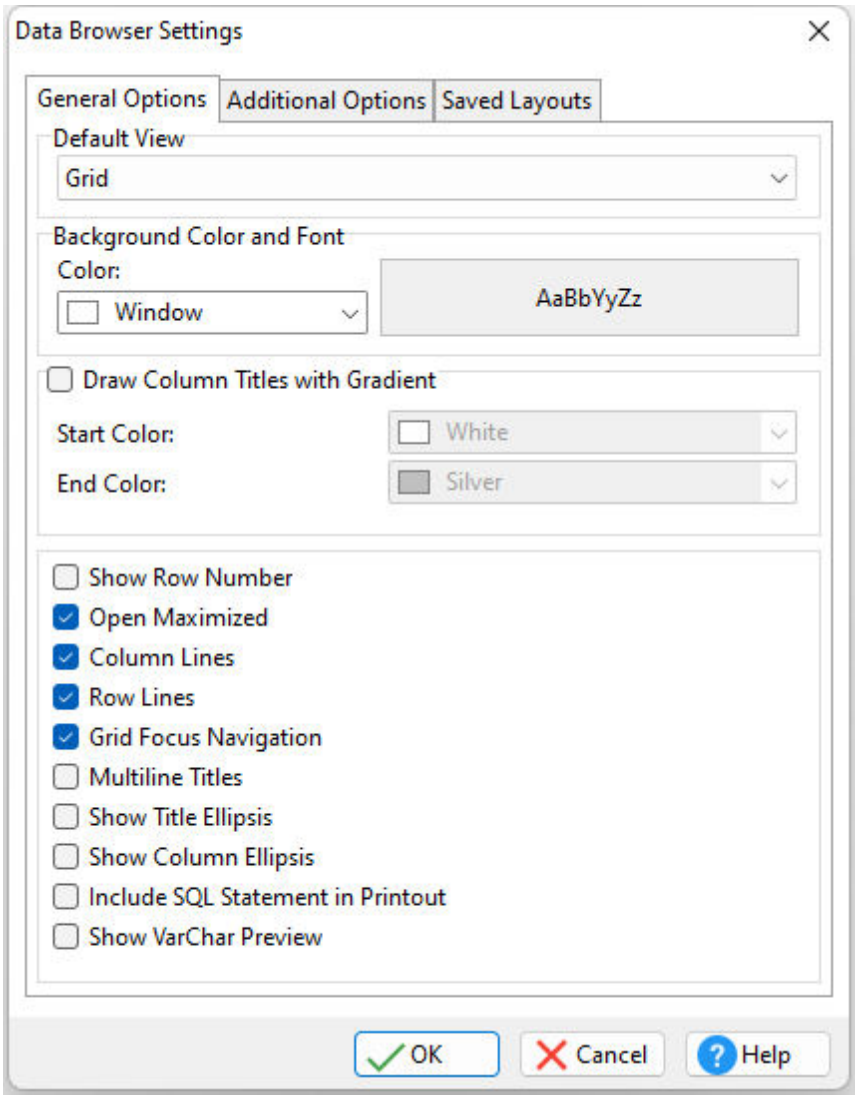
⇨ **Show Title Ellipsis**
Suppresses the extra words of a long title value with "..."

⇨ **Show Column Ellipsis**
Suppresses the extra words or sentences of a long column value with "..."

⇨ **Include SQL Statement in Printout**
Includes the SQL statement with output printed from the Data Browser

⇨ **Show VarChar Preview**
Specifies whether VarChar data is displayed in the Data Browser as an icon (default) or as raw rich text characters

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.5.2 Additional Options

⇨ **Cell Hint Options**
Background Color - specifies the background color of hints
AaBbYyZz - specifies the font style, size and color of hints

⇨ **Selected Row Background and Font Color**
Background color - specifies the background color for the selected row
Text Color - specifies the font color for the selected row

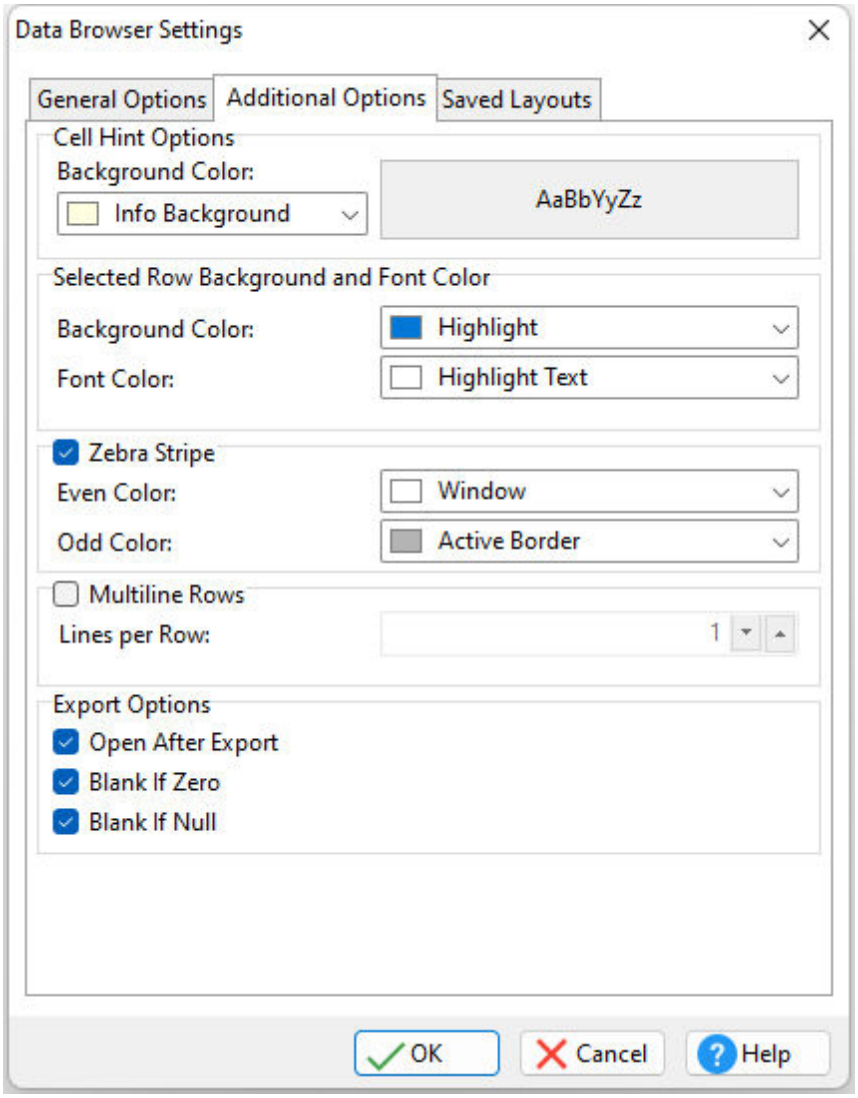
⇨ **Zebra Stripe**
Enables the zebra stripe color effect
Even Color - sets the background color for the even numbered rows
Odd Color - sets the background color for the odd numbered rows

⇨ **Multiline Rows**
Lines per row
Expands the column cell height for additional lines

⇨ **Export Options**

Copyright © 1982-2024 R:BASE Technologies, Inc.

Open After Export - launches the file created based on the current file associated program, when the export is complete
Blank If Zero - exports zero values as blank fields
Blank If Null - exports NULL values as blank fields



2.5.3 Saved Layouts

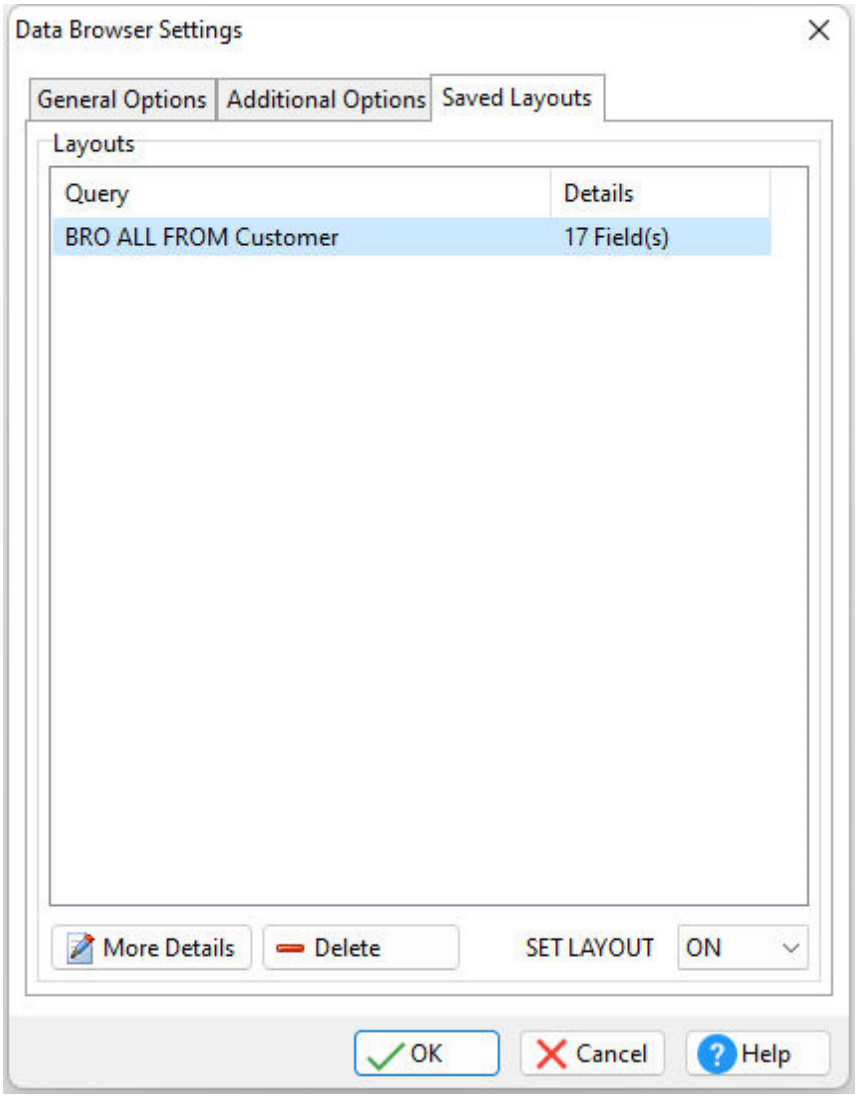
⇨ **Layouts**
Query- displays the query syntax for the layout
Details - displays the number of fields for the layout

More Details - displays the layout details for review
Delete - deletes the selected layout
SET LAYOUT - toggles the layout setting for the Data Browser, to display the custom formatting when viewing tables and views

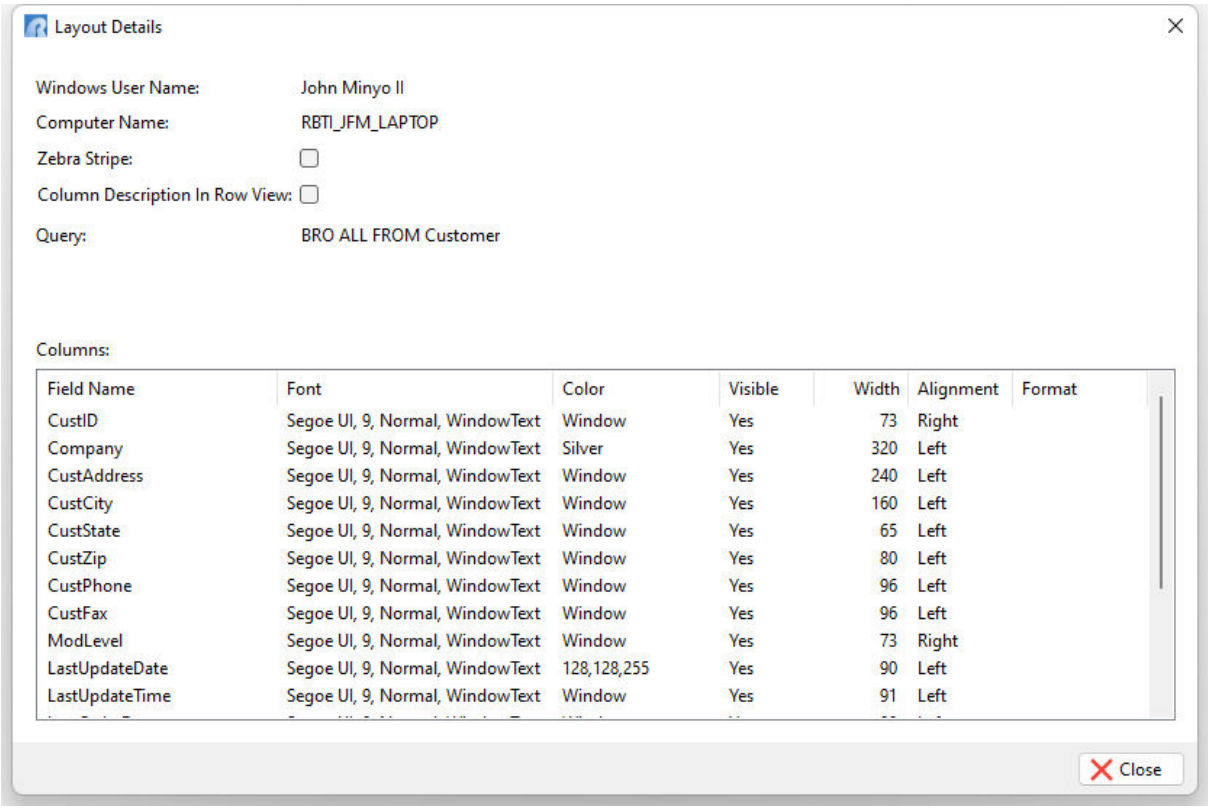
Notes:

- Double clicking a layout will execute the query, where the layout can be previewed in the Data Browser.
- The Saved Layouts tab is only displayed when connected to a database

Copyright © 1982-2024 R:BASE Technologies, Inc.

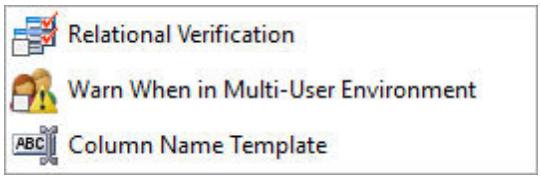


Copyright © 1982-2024 R:BASE Technologies, Inc.



2.6 Data Designer

Relational Verification - allows you to verify the table relation integrity. If checked (true), the relational verification of data type for matching column name will be validated while in the Data Designer, not when saving the table. The default is set to OFF. If un-checked (false), the relational verification will be performed when saving the table.
Warn When in Multi-User Environment - displays a warning when the Data Designer is launched and other users are connected to the database
Column Name Templates - provides a default value when adding new columns to tables



2.7 Form Designer...

The Default Form Settings for the R:BASE Form Designer can be used to assign default settings for all new objects added to your forms.

2.7.1 Form

⇨ **Default Size**
Width
Default: 640

Copyright © 1982-2024 R:BASE Technologies, Inc.

Specifies the horizontal size of the form in pixels. Use the Width property to change the width of the form.
Height
Default: 480
Specifies the vertical size of the form in pixels. Use the Height property to change the height of the form.
Align
Specifies the alignment of the form. The options are:
None - Form can be moved anywhere because it is not aligned to anything
Client - Aligns itself to the available client area by expanding to fill the area that it is in
Left - Aligns itself to the left side of the area growing or shrinking to match parent height
Right - Aligns itself to the right side of the area growing or shrinking to match parent height
Top - Aligns itself to the top side of the area growing or shrinking to match parent width
Bottom - Aligns itself to the bottom side of the area growing or shrinking to match parent width
Auto Scroll
Indicates whether scroll bars appear automatically on the scrolling windowed control if it is not large enough to display all of its controls.

⇨ **Default Font and Color**
Form Color
Default: Button Face
Specifies the background color of the form. Use Color property to read or change the background color of the form.

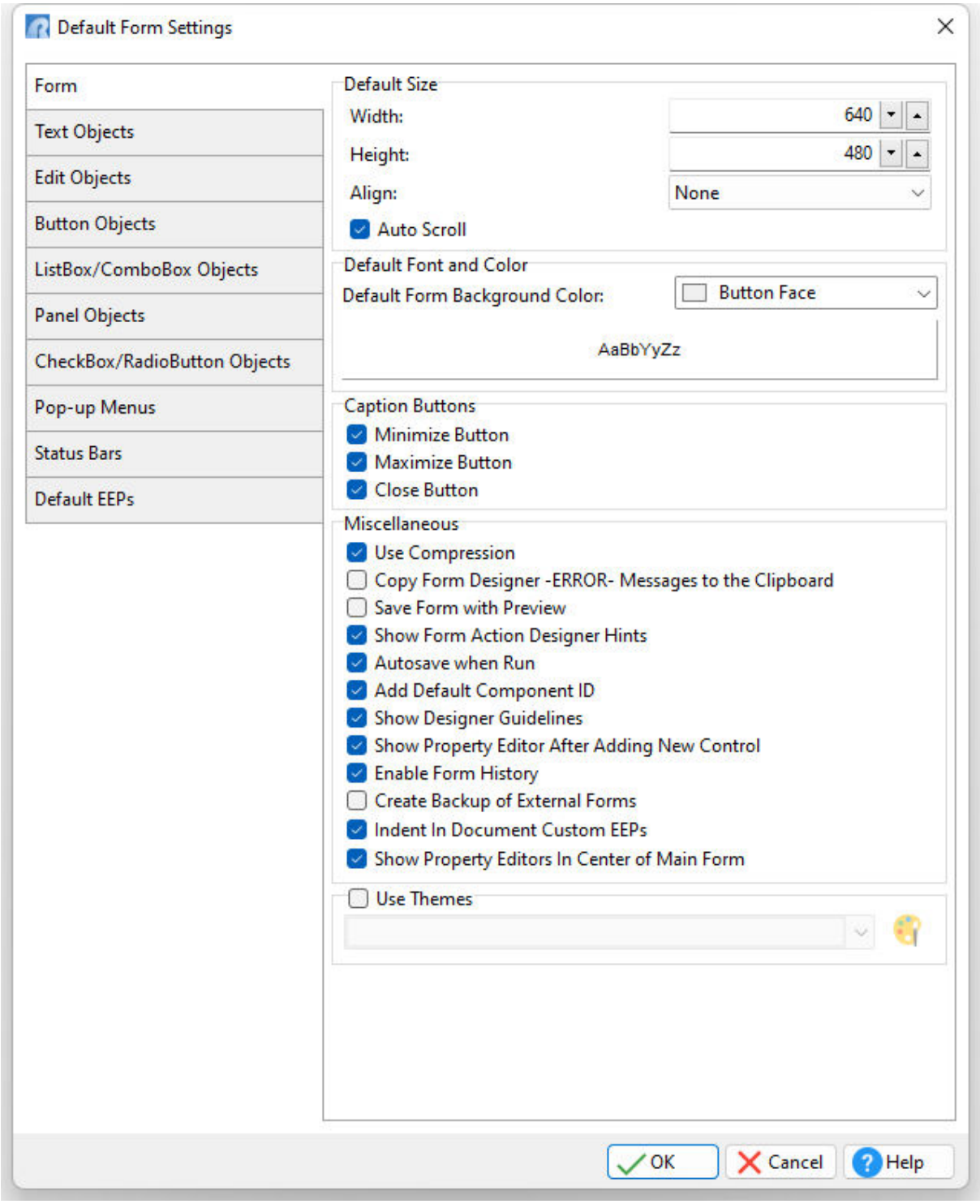
⇨ **Caption Buttons**
Specifies which form caption buttons are visible
Minimize Button
Toggles whether the Minimize Button is visible
Maximize Button
Toggles whether the Maximize Button is visible
Close Button
Toggles whether the Close Button is visible

⇨ **Miscellaneous**
Use Compression
Toggles whether compression is used to minimize space
Copy Form Designer -ERROR- Messages to the Clipboard
Toggles whether occurring error messages are sent to the Windows Clipboard
Save Form with Preview
Allows you preview your forms from within the "Database Explorer" > "Forms" work area, after they are saved. To do so, select a form, right click on the name, and choose "Preview". If the preview is not visible, open the form, change any aspect and save it. The preview will then display.
Show Form Action Designer Hints
Allows you to see the Form Action Designer hints. The Form Action Designer is available from the Menu Bar under "Layout" > "Custom Form Actions".
Autosave when Run
Specifies that R:BASE will save the form when "Run Form" is selected within the Form Designer
Add Default Component ID
Allows R:BASE to add a default Component ID value for new controls added to forms
Show Designer Guidelines
Allows the display of guidelines when dragging/moving one or more selected objects, allowing easier placement of objects to align with existing objects
Show Property Editor After Adding New Control
Specifies if the property dialog appears when adding new controls
Enable Form History
Specifies if R:BASE saves form backups
Create Backup of External Forms
Saves a backup copy of external form files (.rff). The backup copy will reside in the same folder as the original file, only the file extension will contain ".rf~".
Indent In Document Custom EEPs
Specifies if code is indented within the Document Custom EEPs output
Show Property Editors In Center of Main Form
Center property editor windows in the main form. The setting is helpful in multi-monitor environments to place the editor dialog with the appropriate R:BASE session. When OFF, the property editor's last position is persisted.

Copyright © 1982-2024 R:BASE Technologies, Inc.

⇨ **Use Themes**

Enables the use of themes for the form. Themes are artistic representations over the data entry/edit form, which enhance the visual display. There are over 80 pre-defined themes available. After selecting the appropriate theme, it will make your form or application themed automatically. And if you want to remove the theme, just remove the check for the "Use Themes" option.



Copyright © 1982-2024 R:BASE Technologies, Inc.

Windows Control Panel Colors:

Default	The default color for the control to which the color is assigned
ScrollBar	Current color for the of scroll bar track
Background	Current background color of the Windows desktop
ActiveCaption	Current color of the title bar of the active window
InactiveCaption	Current color of the title bar of inactive windows
Menu	Current background color of menus
Window	Current background color of windows
WindowFrame	Current color of window frames
WindowText	Current color of text in windows
CaptionText	Current color of the text on the title bar of the active window
ActiveBorder	Current border color of the active window
InactiveBorder	Current border color of inactive windows
AppWorkSpace	Current color of the application workspace
Highlight	Current background color of selected text
HightlightText	Current color of selected text
BtnFace	Current color of a button face
BtnShadow	Current color of a shadow cast by a button
GrayText	Current color of text that is dimmed
BtnText	Current color of text on a button
InactiveCaptionText	Current color of the text on the title bar of an inactive window
BtnHighlight	Current color of the highlighting on a button
3DDkShadow	Dark shadow for three-dimensional display elements
3DLight	Light color for three-dimensional display elements (for edges facing the light source)
InfoText	Text color for tool tip controls
InfoBk	Background color for tool tip controls

2.7.2 Text Objects

⇨ **Background Font, Color and Autosize**

Color
Specifies the background color of the object
Auto Size
Object will shrink or stretch automatically based on value displayed
Use Parent's Font
Specifies to use the font for the control's parent object/control
AaBbYyZz
Specifies the font style, size and color for the object

⇨ **Border Sides**

Specifies which sides of the border are visible
Border Color
Specifies the color of the object's border
Border Highlight Color
Specifies the highlight color of the object's border
Border Inner
Specifies the border style for the inner portion of the border
Border Outer
Specifies the border style for the outer portion of the border
Border Shadow
Specifies the shadow color of the object's border

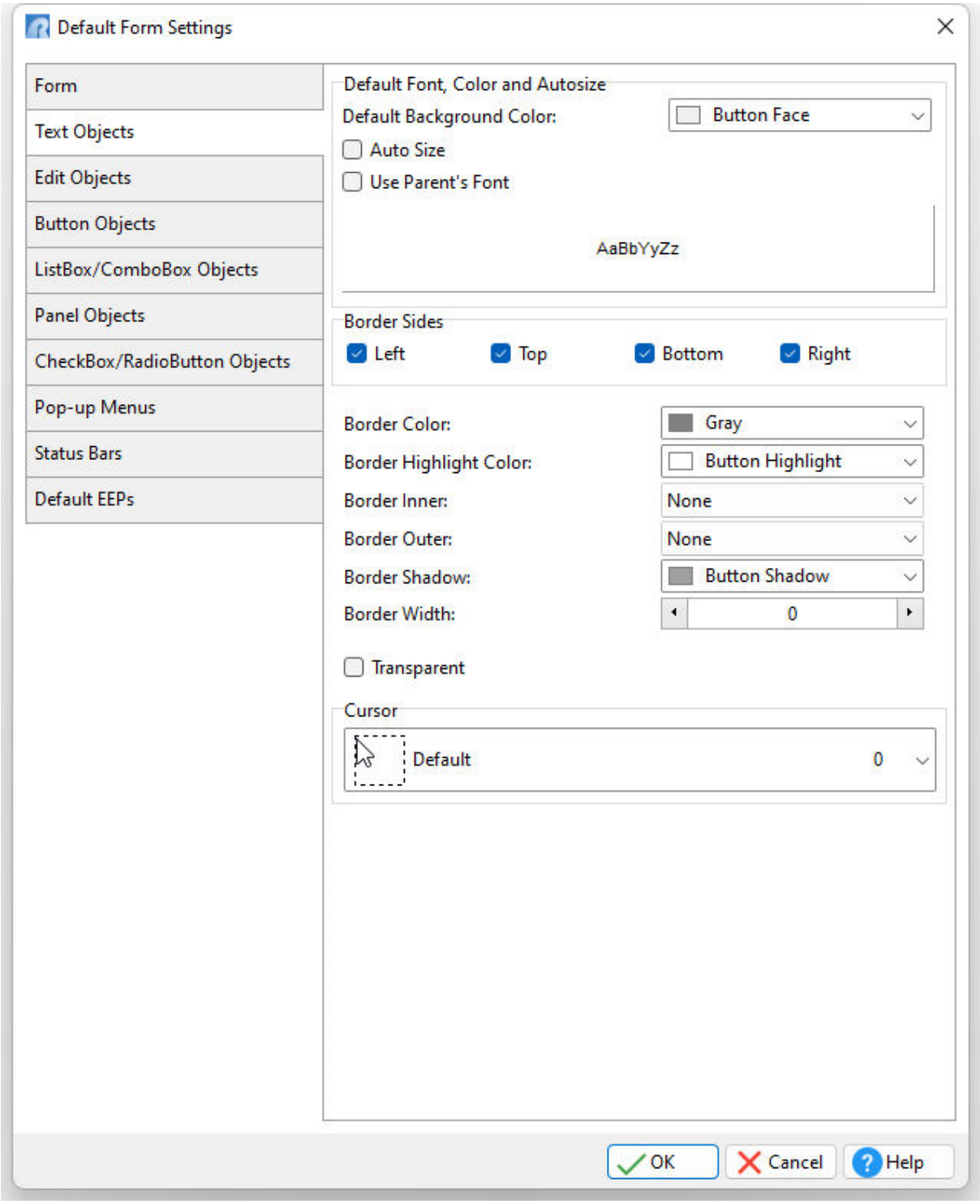
Copyright © 1982-2024 R:BASE Technologies, Inc.

Border Width
Specifies the width of the object's border

Transparent
Allows the background to become transparent to the parent object

⇨ **Cursor**
Specifies the visual representation of the mouse cursor as it hovers over any of your text objects.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.7.3 Edit Objects

⇨ **Default Font and Color**

Color

Specifies the background color of the object

Copyright © 1982-2024 R:BASE Technologies, Inc.

Use Parent's Font
Specifies to use the font for the control's parent object/control
AaBbYyZz
Specifies the font style, size and color for the object

⇨ **Visible Frame**
Displays object frame when checked

⇨ **Frame Hot Track**
Enables the frame to glow when the mouse enters the control

⇨ **Windows Theme**
Applies the current Windows theme to the control

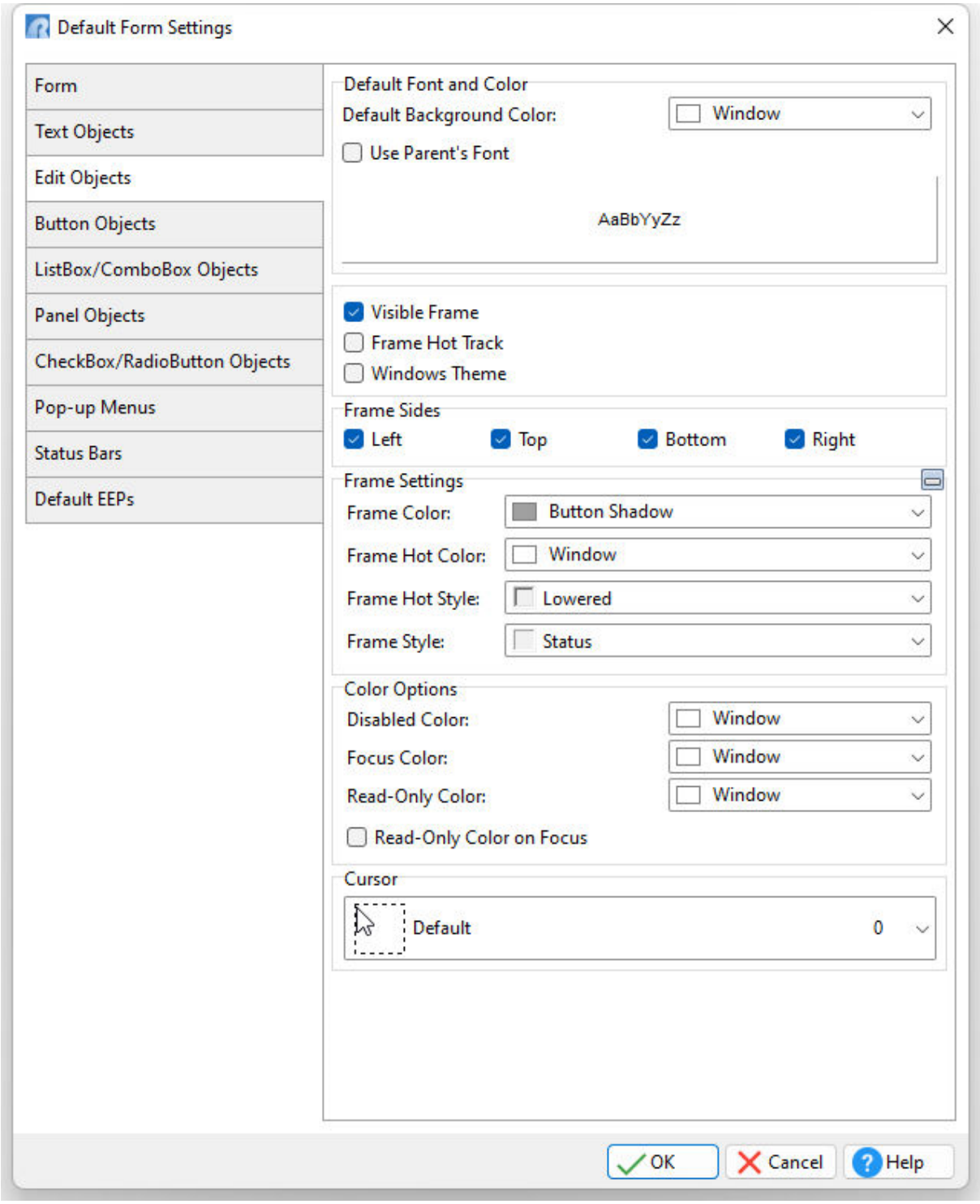
⇨ **Frame Sides**
Specifies which sides of the border are visible

⇨ **Frame Settings**
Frame Color
Specifies the color of the frame
Frame Hot Color
Specifies the glow color of the frame when "Hot Track" is enabled
Frame Hot Style
Specifies the hot style of the frame when "Hot Track" is enabled
Frame Style
Specifies the style of the frame

⇨ **Color Options**
Disabled Color
Specifies the background color of the control when disabled
Focus Color
Specifies the background color of the control when focused
Read-Only Color
Specifies the background color of the control when set to "read only"
Read-Only Color on Focus
Enables the "Read-Only" Color if the cursor focus is on the control

⇨ **Cursor**
Specifies the visual representation of the mouse cursor as it hovers over any of your edit objects.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.7.4 Button Objects

⇨ **Default Font and Color**

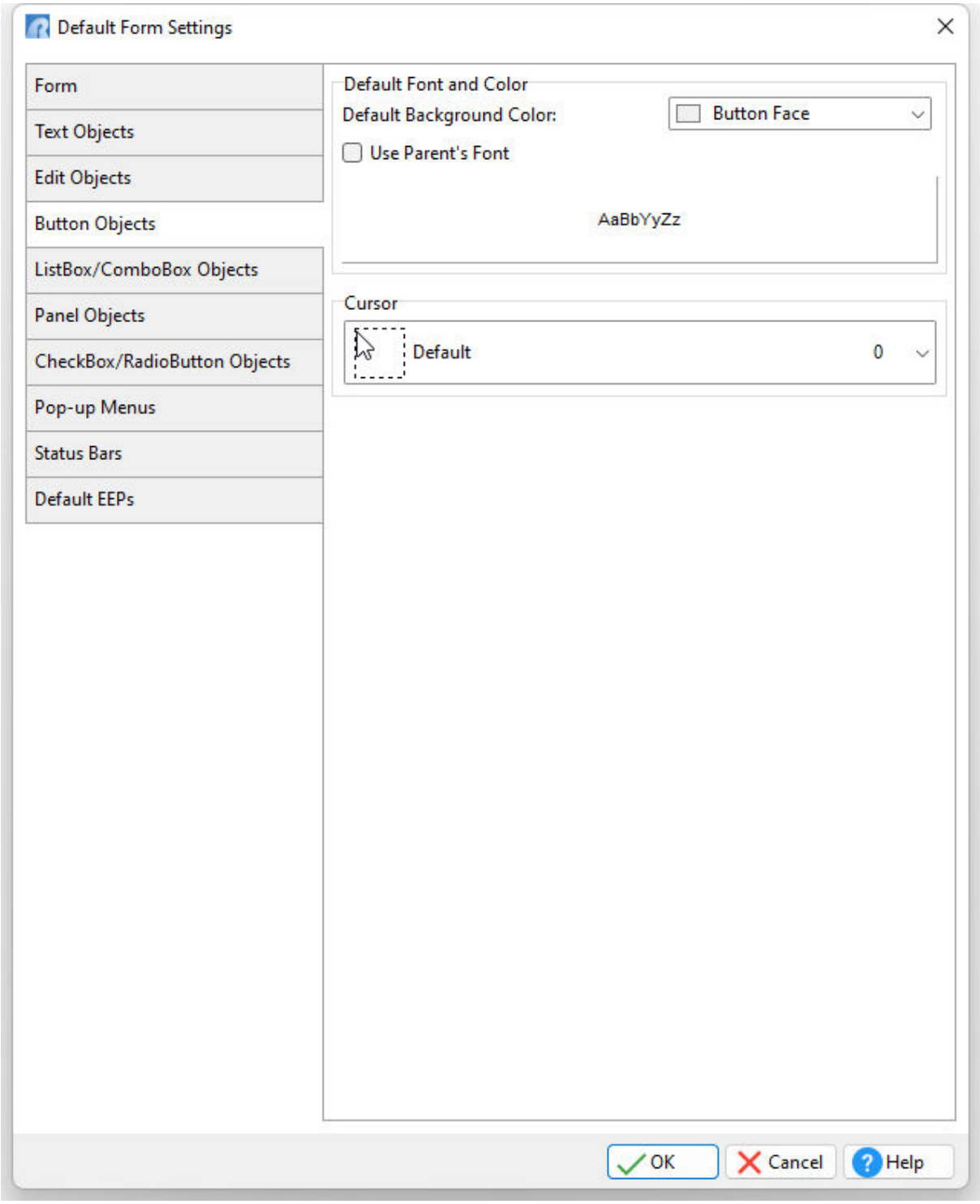
Color

Specifies the background color of the object

Copyright © 1982-2024 R:BASE Technologies, Inc.

Use Parent's Font
Specifies to use the font for the control's parent object/control
AaBbYyZz
Specifies the font style, size and color for the object

⇨ **Cursor**
Specifies the visual representation of the mouse cursor as it hovers over any of your button objects.



Copyright © 1982-2024 R:BASE Technologies, Inc.

2.7.5 ListBox/ComboBox Objects

⇨ **Default Font and Color**

Color
Specifies the background color of the object
Use Parent's Font
Specifies to use the font for the control's parent object/control
AaBbYyZz
Specifies the font style, size and color for the object

Visible Frame
Displays object frame when checked
Flat Frame
Displays object frame as a flat frame when checked and Visible Frame is checked

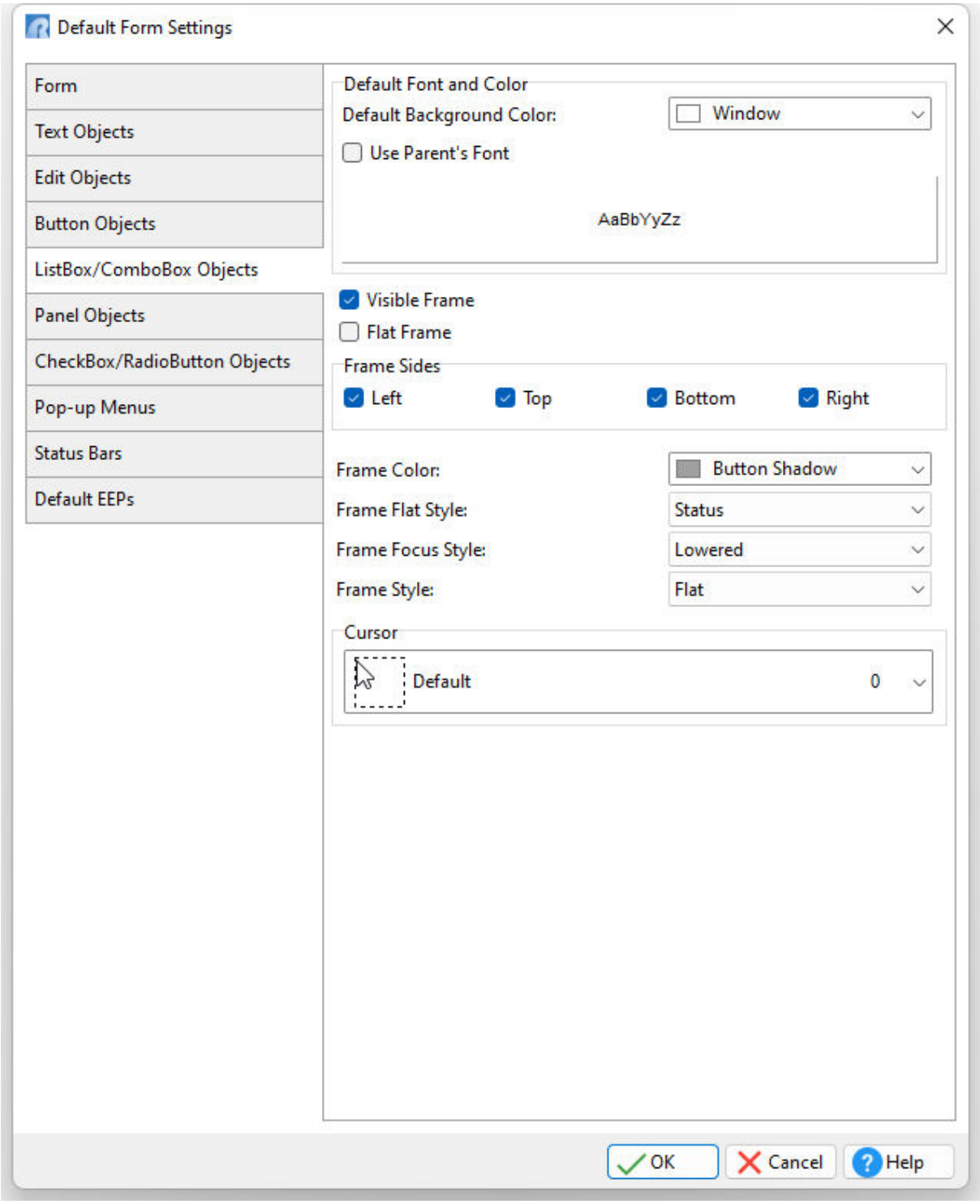
⇨ **Frame Sides**

Specifies which sides of the border are visible
Frame Color
Specifies the color of the object's frame
Frame Flat Style
Specifies the style of the object's frame when Flat Frame is checked
Frame Focus Style
Specifies the style of the object's frame when focus is on the object
Frame Style
Specifies the style of the object's frame when Flat Frame is unchecked and Visible Frame is checked

⇨ **Cursor**

Specifies the visual representation of the mouse cursor as it hovers over any of your ListBox/ComboBox objects.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.7.6 Panel Objects

⇨ **Default Font and Color**

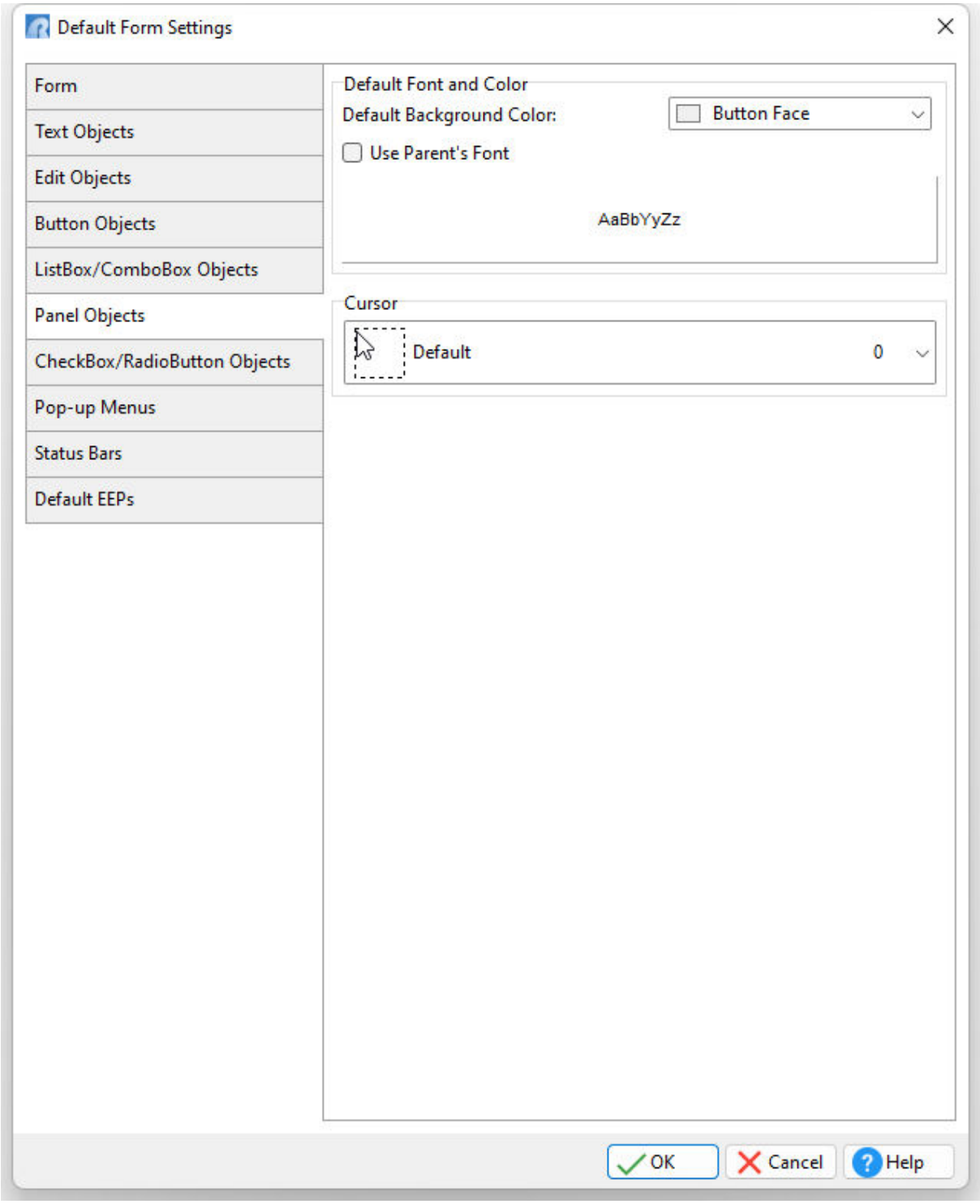
Color

Specifies the background color of the object

Copyright © 1982-2024 R:BASE Technologies, Inc.

Use Parent's Font
Specifies to use the font for the control's parent object/control
AaBbYyZz
Specifies the font style, size and color for the object

⇨ **Cursor**
Specifies the visual representation of the mouse cursor as it hovers over any of your panel objects.



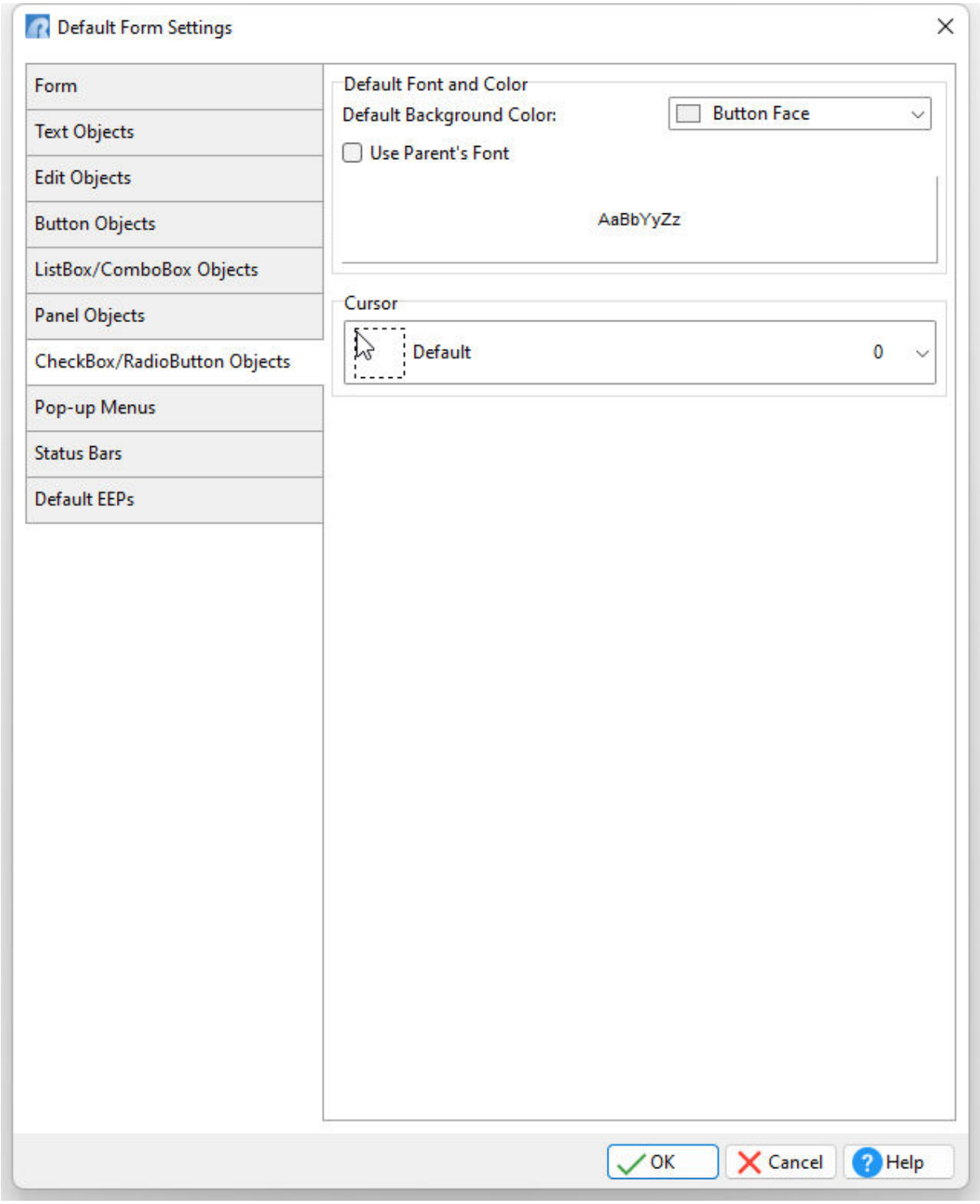
Copyright © 1982-2024 R:BASE Technologies, Inc.

2.7.7 CheckBox/RadioButton Objects

⇨ **Default Font and Color**
Color
Specifies the background color of the object
Use Parent's Font
Specifies to use the font for the control's parent object/control
AaBbYyZz
Specifies the font style, size and color for the object

⇨ **Cursor**
Specifies the visual representation of the mouse cursor as it hovers over any of your CheckBox/RadioButton objects.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.7.8 Pop-up Menus

⇨ **Lines**
Number of Lines
Limits the number of values displayed in the Pop-up Menu

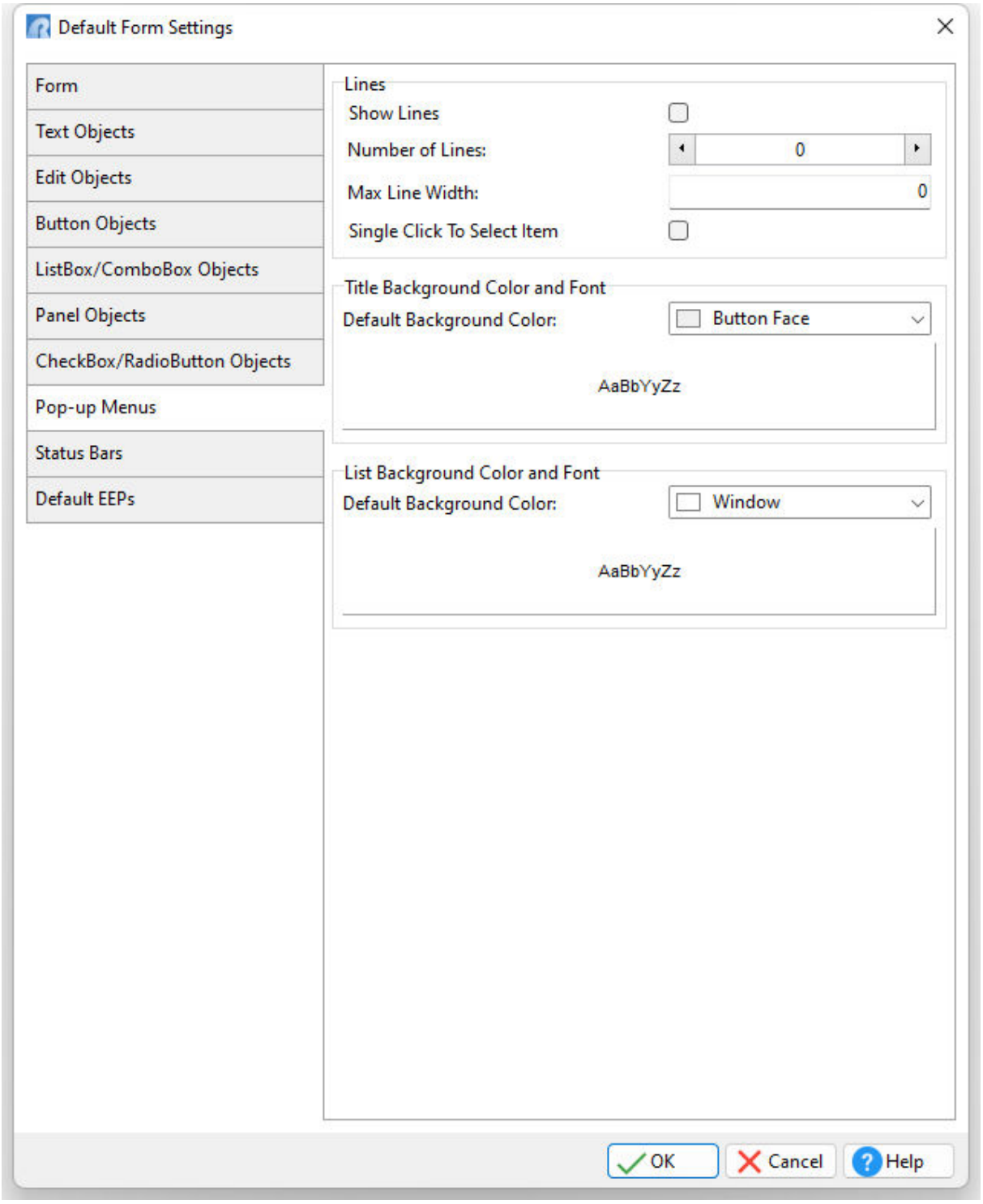
Copyright © 1982-2024 R:BASE Technologies, Inc.

Show Lines
Displayed lines between the values in the Pop-up Menu
Max Line Width
Limits the width of the Pop-up Menu
Single Click To Select Item
Specifies if just one click is accepted to select an item and close the Pop-up Menu

⇨ **Title Background Color and Font**
Specifies the background color of the title
AaBbYyZz
Specifies the font style, size and color for the title

⇨ **List Background Color and Font**
Specifies the background color of the list
AaBbYyZz
Specifies the font style, size and color for the list

Copyright © 1982-2024 R:BASE Technologies, Inc.

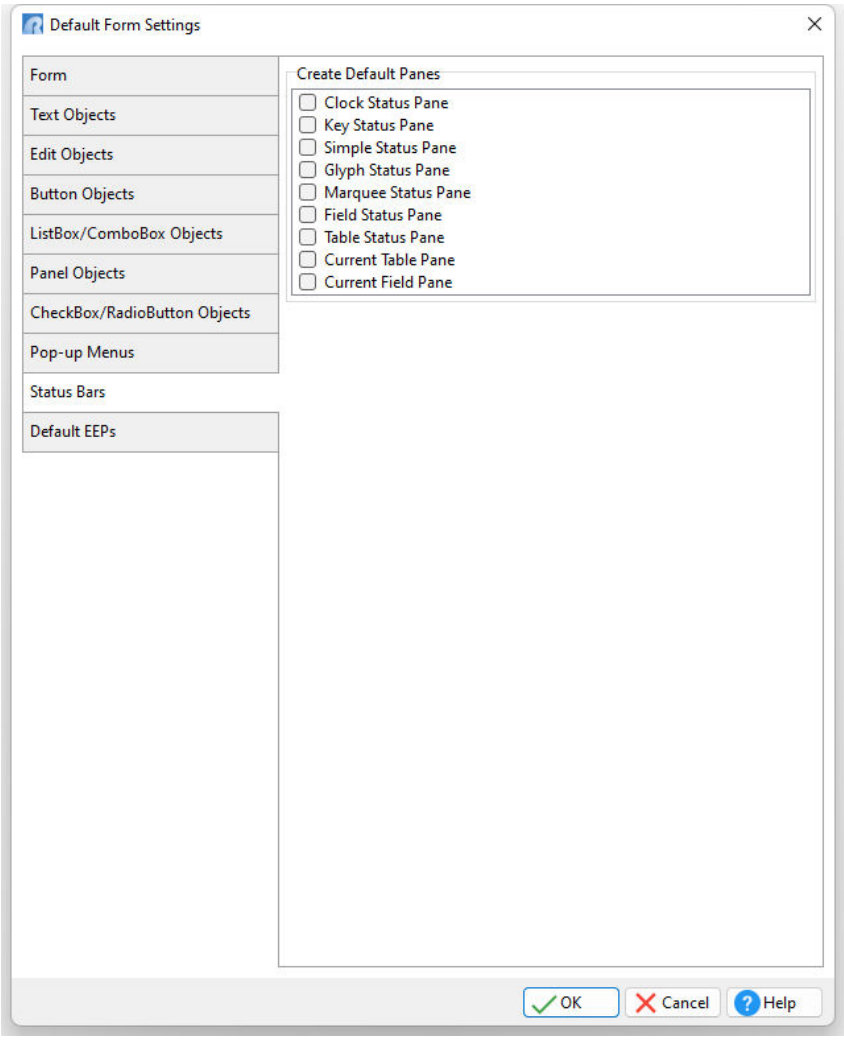


Copyright © 1982-2024 R:BASE Technologies, Inc.

2.7.9 Status Bars

Create Default Panes
Specifies which pane is added to the Status Bar by default. Options include:

- Clock Status Pane
- Key Status Pane
- Simple Status Pane
- Glyph Status Pane
- Marquee Status Pane
- Table Status Pane
- Field Status Pane
- Current Table Pane
- Current Field Pane

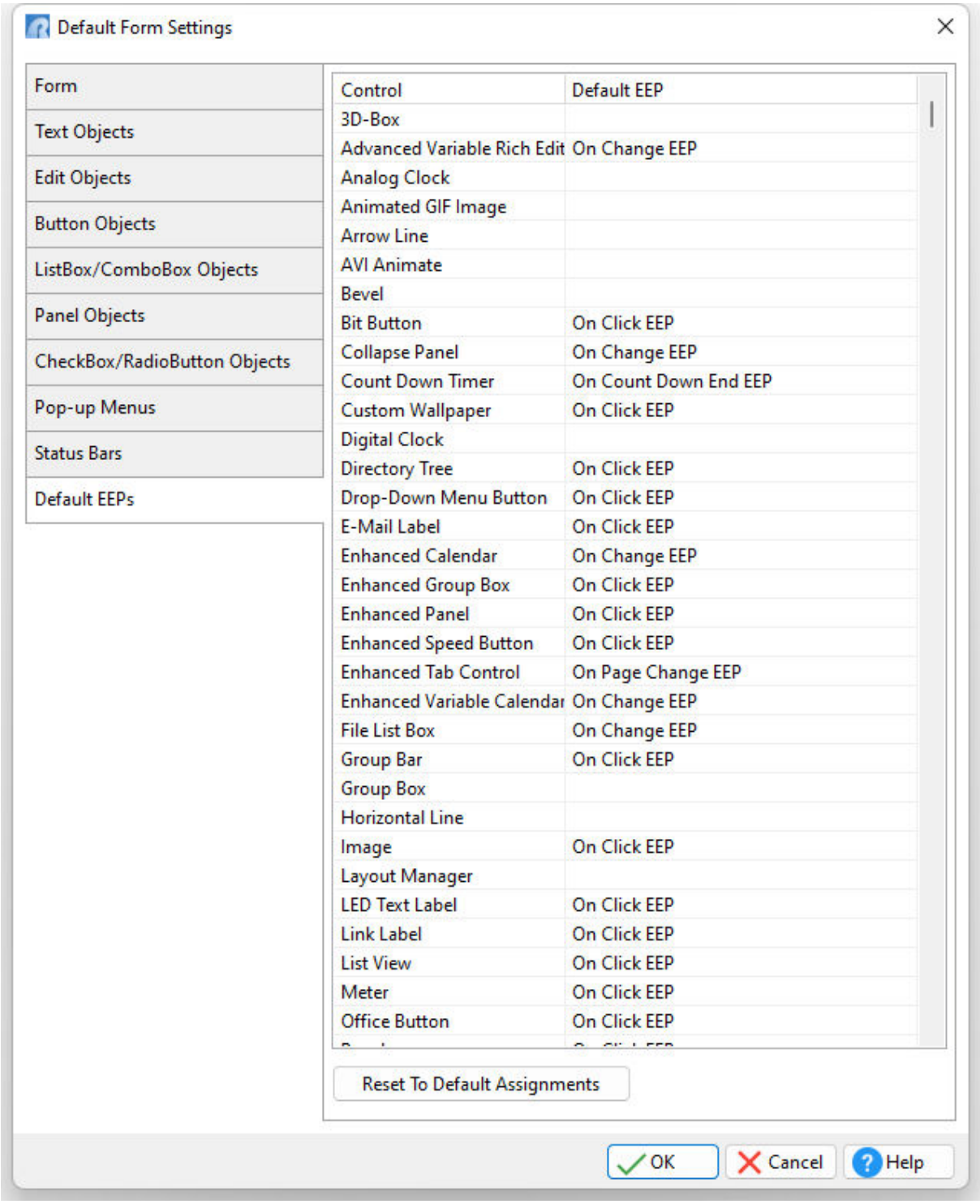


2.7.10 Default EEPs

The options allow users to edit the default EEP assignment for form controls.

Modifying the default EEP assignment helpful when using [Ctrl]+double click to launch the R:BASE Editor to modify a control's default EEP command syntax.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.8 Report/Label Designer

The Default Settings for the R:BASE Report Designer and Label Designer can be used to assign default settings for all new objects added to your reports and labels, the printer, as well as the paper size and source.

Copyright © 1982-2024 R:BASE Technologies, Inc.

The Report E-Mail Settings are used to assign your SMTP settings when sending reports as email attachments after the report is generated.

2.8.1 Default Settings

The Default Report Settings for the R:BASE Report Designer can be used to assign default settings for all new objects added to your reports, the printer, as well as the paper size and source.

2.8.1.1 General

⇨ **Row Count for Preview**
Sets the number of rows to be displayed when previewing a report/label

⇨ **Copy Report/Label -ERROR- Messages to the Clipboard**
Toggles whether occurring error messages are sent to the Windows Clipboard

⇨ **Add Default Component ID**
Add a default Component ID value for new controls added to reports and labels

⇨ **Autosave When Preview**
Saves the report/label changes within the Designer when previewed

⇨ **Compress Reports/Labels**
Toggles whether compression is used to minimize space

⇨ **Enable Report/Label History**
Specifies if R:BASE saves report/label backups

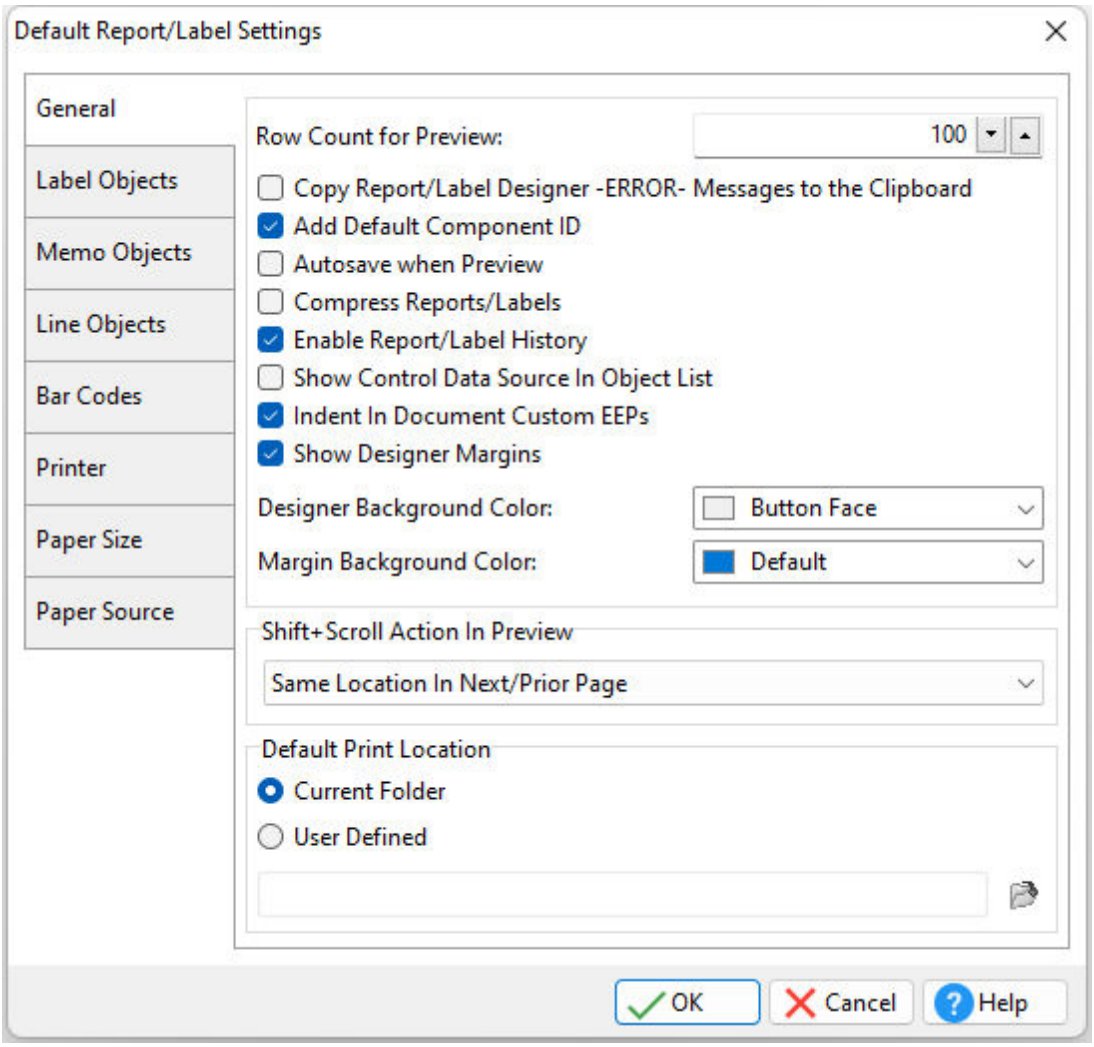
⇨ **Show Control Data Source In Object List**
Specifies to show the source name of each variable and database object within the Object List toolbar

⇨ **Designer Background Color**
Sets the background color for the Report/Label Designer

⇨ **Shift+Scroll Action in Preview**
Sets the scroll behavior when moving from page to page in the Print Preview screen

⇨ **Default Print Location**
Sets the current folder or a defined location for print output

Copyright © 1982-2024 R:BASE Technologies, Inc.

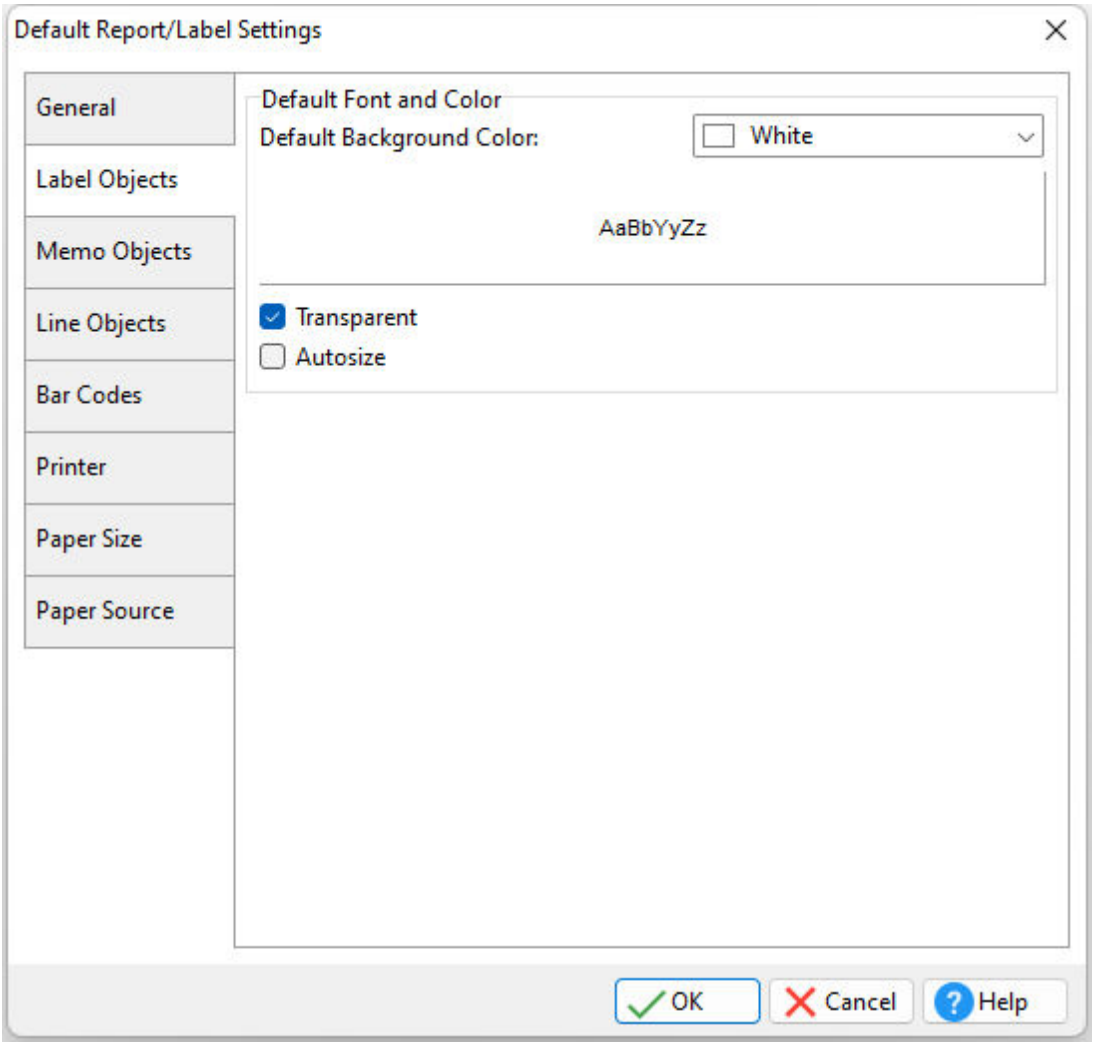


2.8.1.2 Label Objects

⇨ **Background Color and Font**

Color
Specifies the background color of the object
AaBbYyZz
Specifies the font style, size and color for the object
Transparent
Allows the object to become transparent to the background object
Auto Size
Object will shrink or stretch automatically based on value displayed

Copyright © 1982-2024 R:BASE Technologies, Inc.

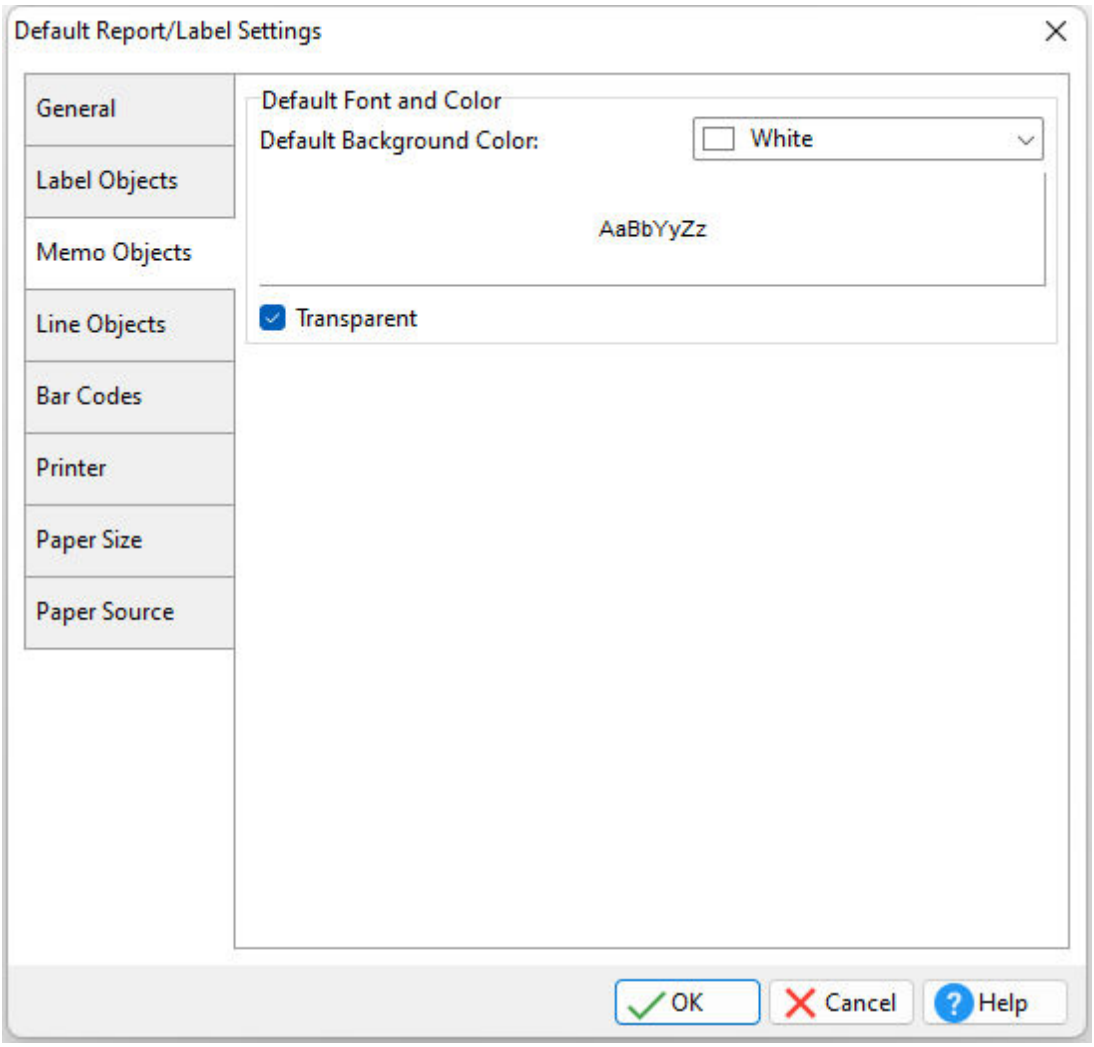


2.8.1.3 Memo Objects

⇨ **Background Color and Font**

Color
Specifies the background color of the object
AaBbYyZz
Specifies the font style, size and color for the object
Transparent
Allows the object to become transparent to the background object

Copyright © 1982-2024 R:BASE Technologies, Inc.

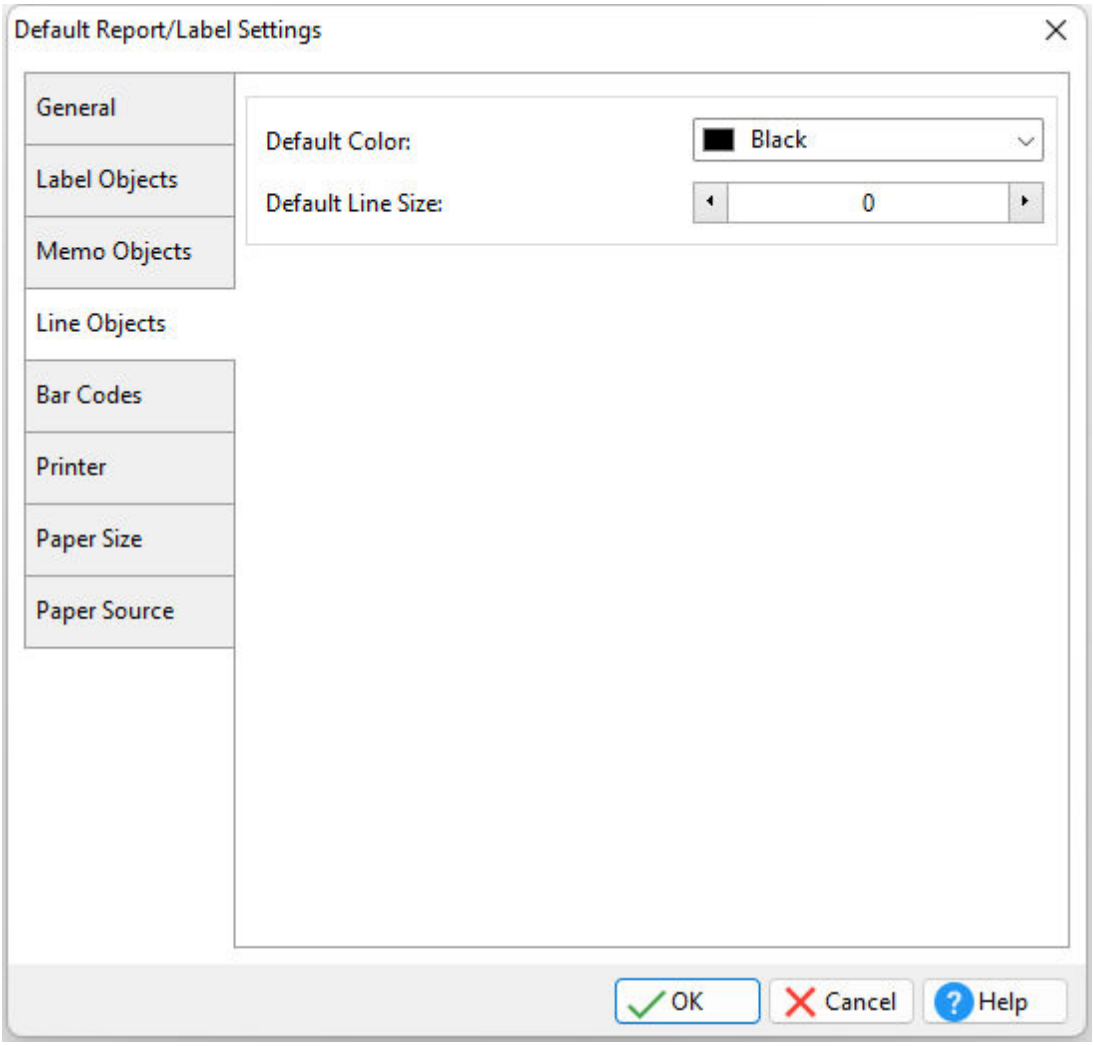


2.8.1.4 Line Objects

⇨ **Default Color**
Specifies the background color of the object

⇨ **Default Line Size**
Specifies the line size of the object

Copyright © 1982-2024 R:BASE Technologies, Inc.



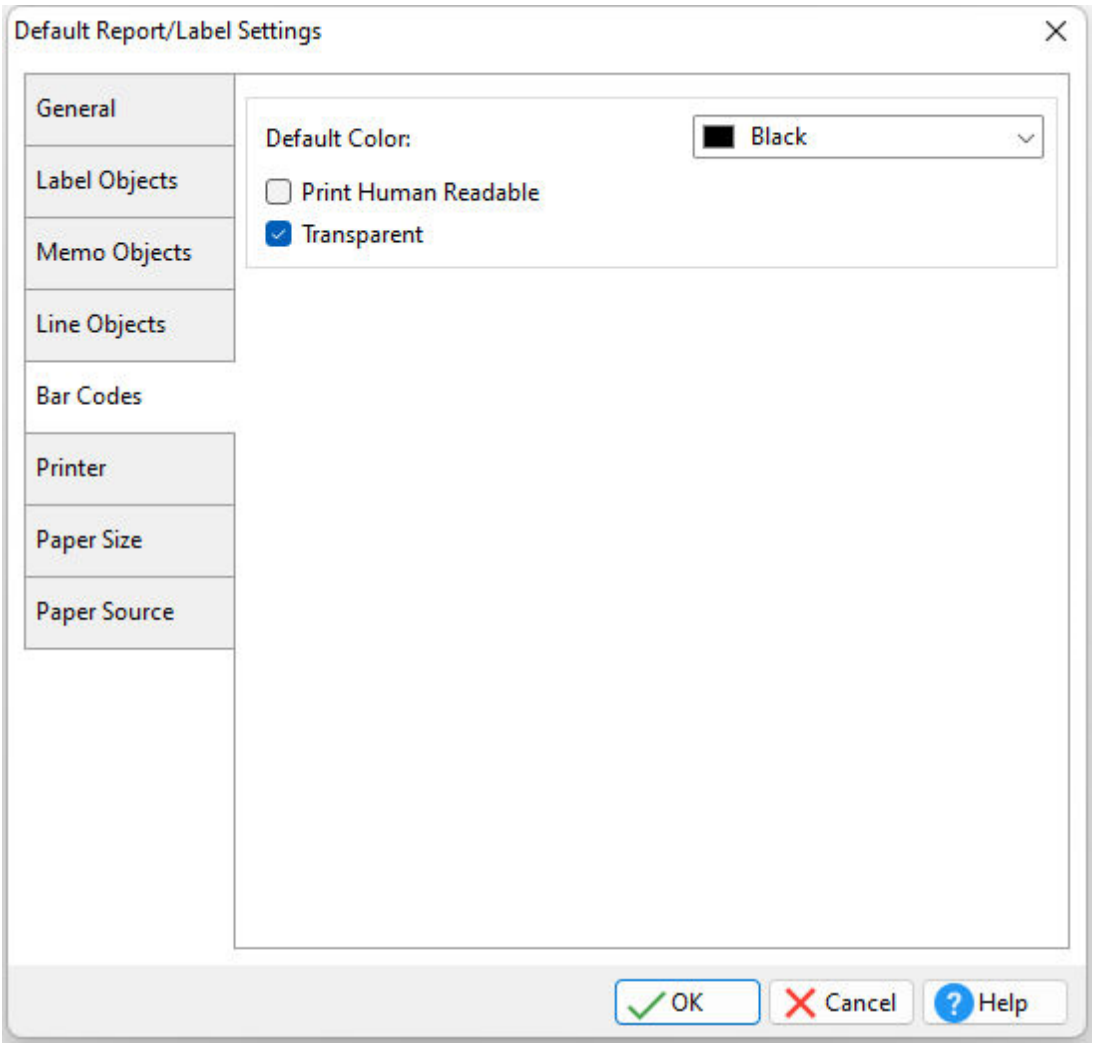
2.8.1.5 Bar Codes

⇨ **Default Color**
Specifies the color of the object

⇨ **Print Human Readable**
Specifies if the data is displayed in human readable format

⇨ **Transparent**
Allows the object to become transparent to the background object

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.8.1.6 Printer

⇨ **Printer Name**
Specifies the printer

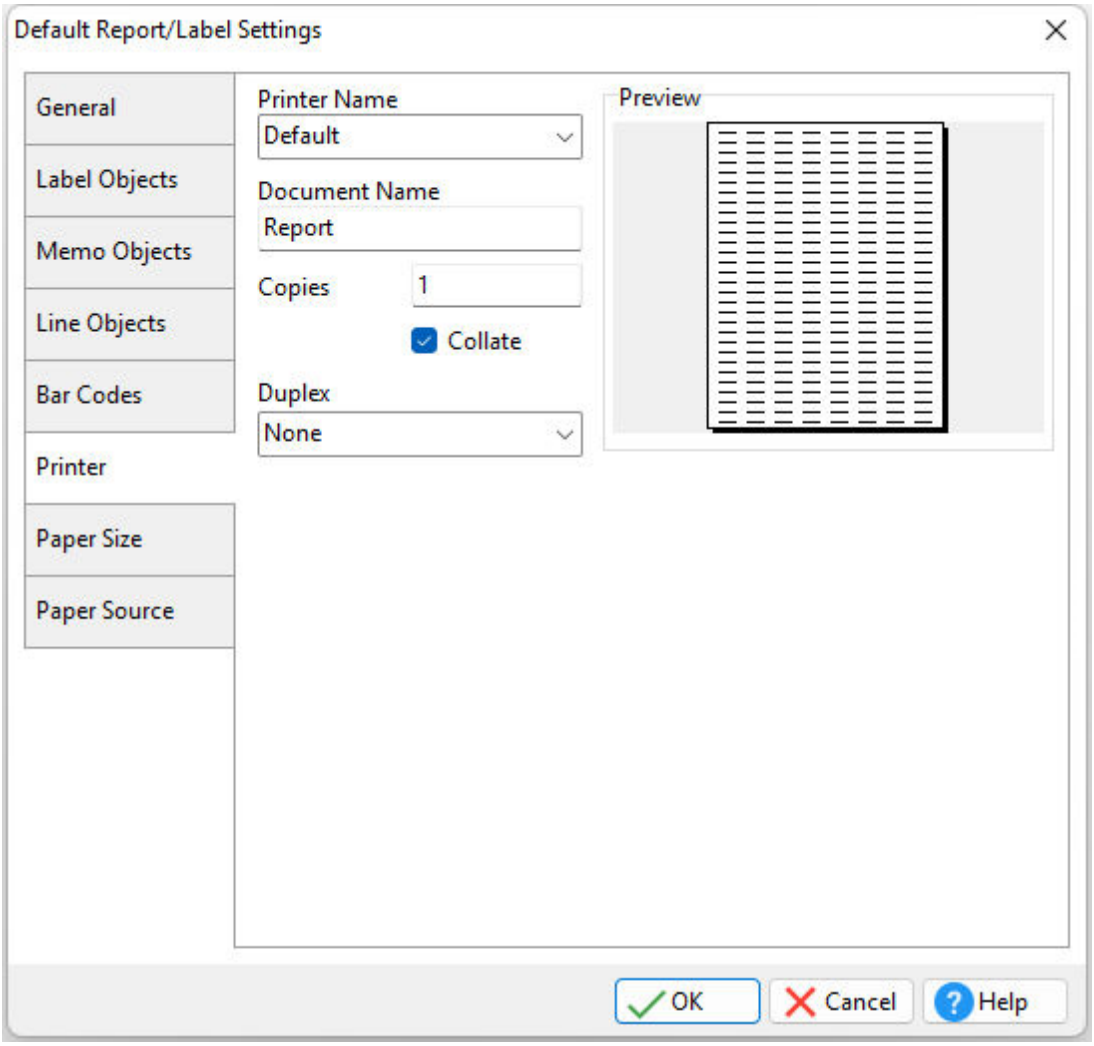
⇨ **Document Name**
Specifies the report name

⇨ **Copies**
Specifies the number of copies

⇨ **Collate**
Specifies whether multiple copies will print in pre-sorted order

⇨ **Duplex**
Specifies what type of two-sided printing should occur

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.8.1.7 Paper Size

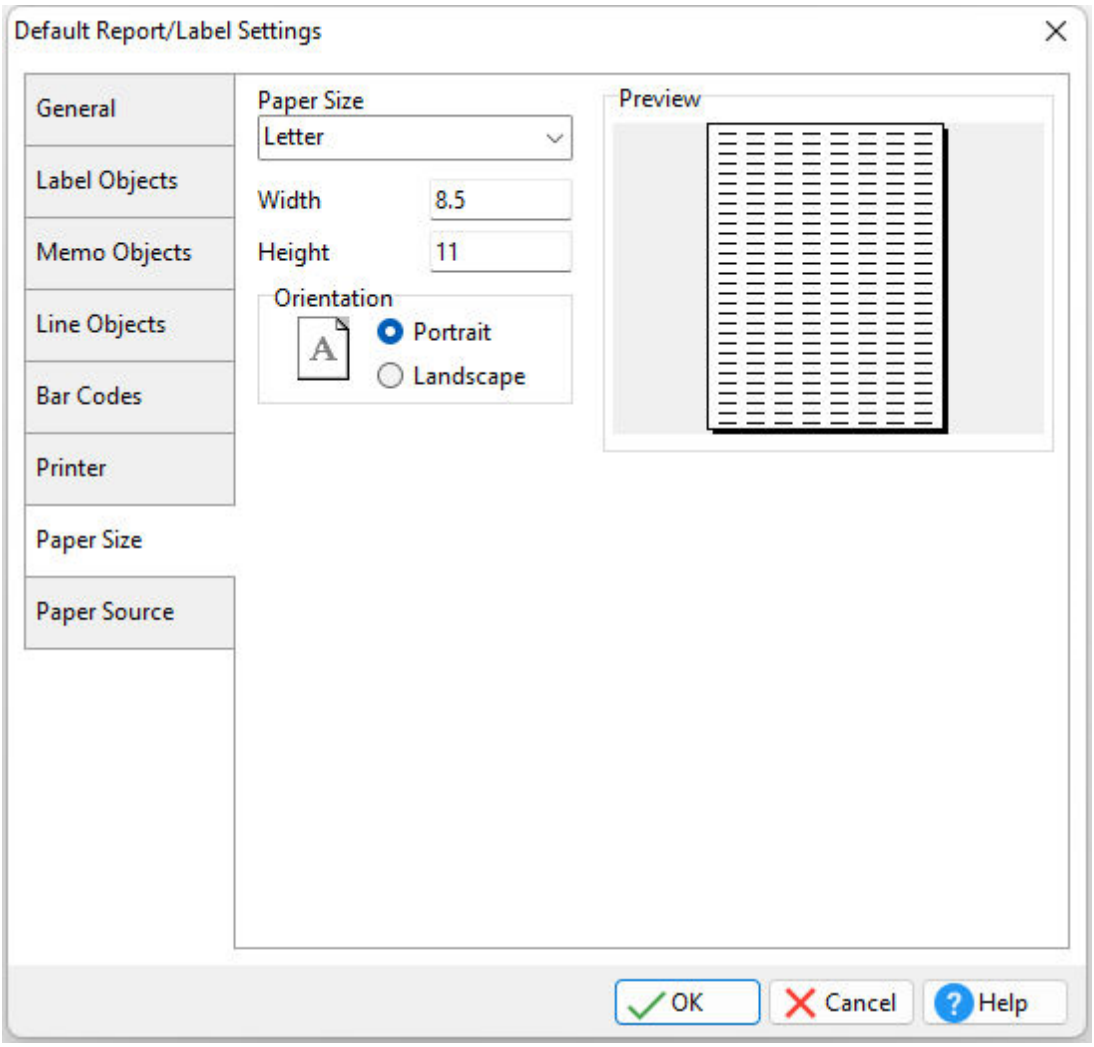
⇨ **Paper Size**
Specifies the paper size

⇨ **Width**
Specifies the paper width

⇨ **Height**
Specifies the paper height

⇨ **Orientation**
Specifies whether a report prints vertically or horizontally on a page

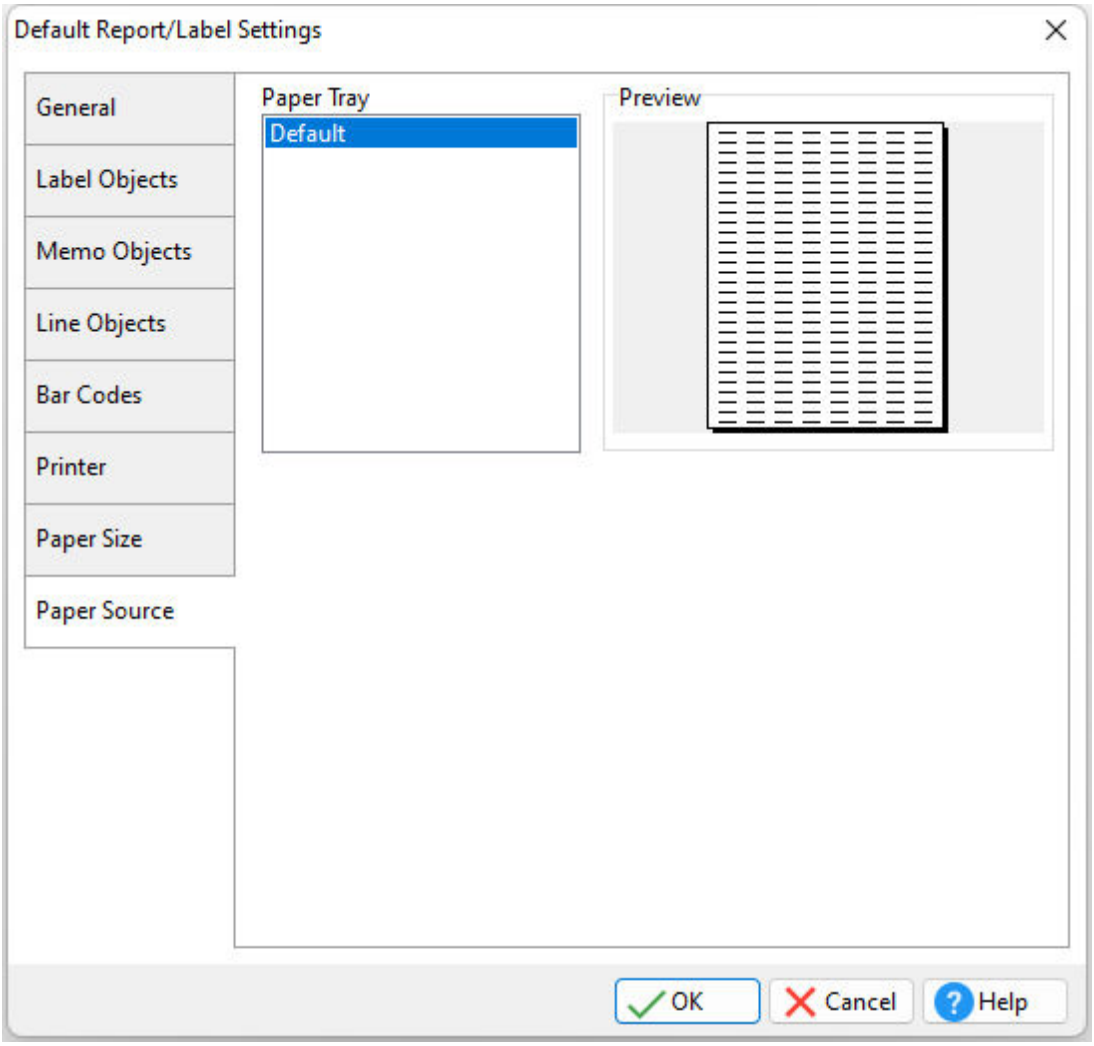
Copyright © 1982-2024 R:BASE Technologies, Inc.



2.8.1.8 Paper Source

⇨ **Paper Tray**
Specifies the name of the bin (paper tray) containing the paper

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.8.2 Report E-Mail Settings...

The Report E-Mail Settings are used to assign your SMTP settings when sending reports as email attachments after the report is generated.

To review the default settings, choose "Settings" > "Report/Label Designer" > "Report E-Mail Settings..." from the Menu Bar.

⇨ **Server Name**
Specifies Hostname/IP Address of SMTP Server (Outgoing). Also sometimes known as your 'Sending Mail Server', your SMTP server receives and processes all of your outgoing mail.

For example, your Outgoing Mail Server might be called something like 'my-internet-provider.com' or 'outgoing-mail.mycompany.com'. The SMTP server name is sometimes the same as your incoming mail server name. If your ISP or your network administrator has not given this server name to you, you can leave the box blank. If you do not know what your outgoing mail server name is, you need to contact your Internet Service Provider or Network Administrator.

⇨ **Port**
Specifies the integer value of SMTP Server Port (Default = 25).

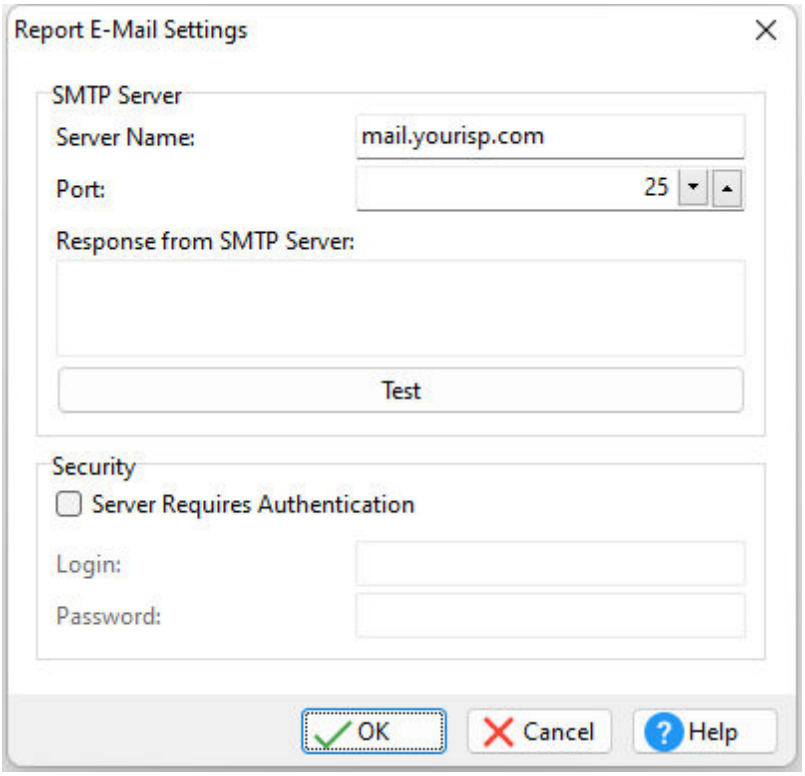
⇨ **Response from SMTP Server**
Displays the response from the SMTP server after the "Test" button is selected.

Copyright © 1982-2024 R:BASE Technologies, Inc.

⇨ **Test**
Verifies the SMTP server settings. You will receive one of two messages back, either "Connection Established!", or "Connection Failed.". If the connection is established, then you will receive a message from the SMTP server. This message will be displayed in the "Response from SMTP Server" box. If the connection fails, then you will need to obtain the proper setting information from either your network administrator or ISP. The "Connection Established!" message does not verify an email will be sent successfully. The security settings may still need configured to send email.

⇨ **Security**

Server Requires Authentication
If the SMTP mail server that you are using requires authentication, you must enable the "Server Requires Authentication" check box and enter your Login Name and Password.
Login
Specifies the SMTP user name
Password
Specifies the SMTP password



2.9 Application Designer

Create Backup Copy - allows you to save a backup copy of the R:BASE application file (.RBA). The backup copy will reside in the same folder as the original file, only the file extension will contain ".rb~".

2.10 BLOB Editor...

The BLOB Editor Settings specifies the default values for the R:BASE BLOB Editor.

Copyright © 1982-2024 R:BASE Technologies, Inc.

2.10.1 Image

⇨ **Default Page**
Specifies the default page displayed when the BLOB Editor is launched for an empty data set. If data exists, the BLOB Editor will default to the appropriate page based on the type of data stored.

⇨ **Default Image Format**
Specifies the default format when saving images

⇨ **Zoom**
Specifies the default zoom when displaying images

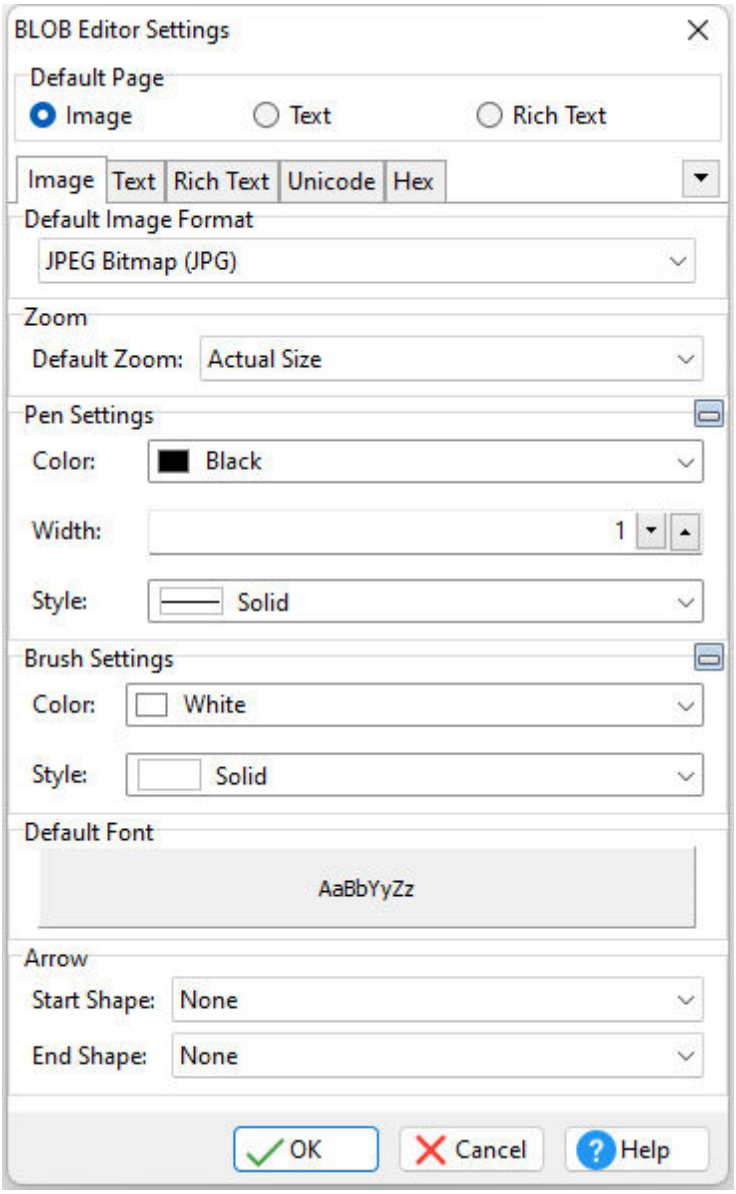
⇨ **Pen Settings**
Color - specifies the line color
Width - specifies the line width
Style - specifies the line style

⇨ **Brush Settings**
Color - specifies the fill color for the arrow start and end shapes, if defined
Style - specifies the fill style for the arrow start and end shapes, if defined

⇨ **Default Font**
Specifies the default text font type, size, and style

⇨ **Arrow**
Start Shape - specifies the arrow start shape for the line
End Shape - specifies the arrow end shape for the line

Copyright © 1982-2024 R:BASE Technologies, Inc.



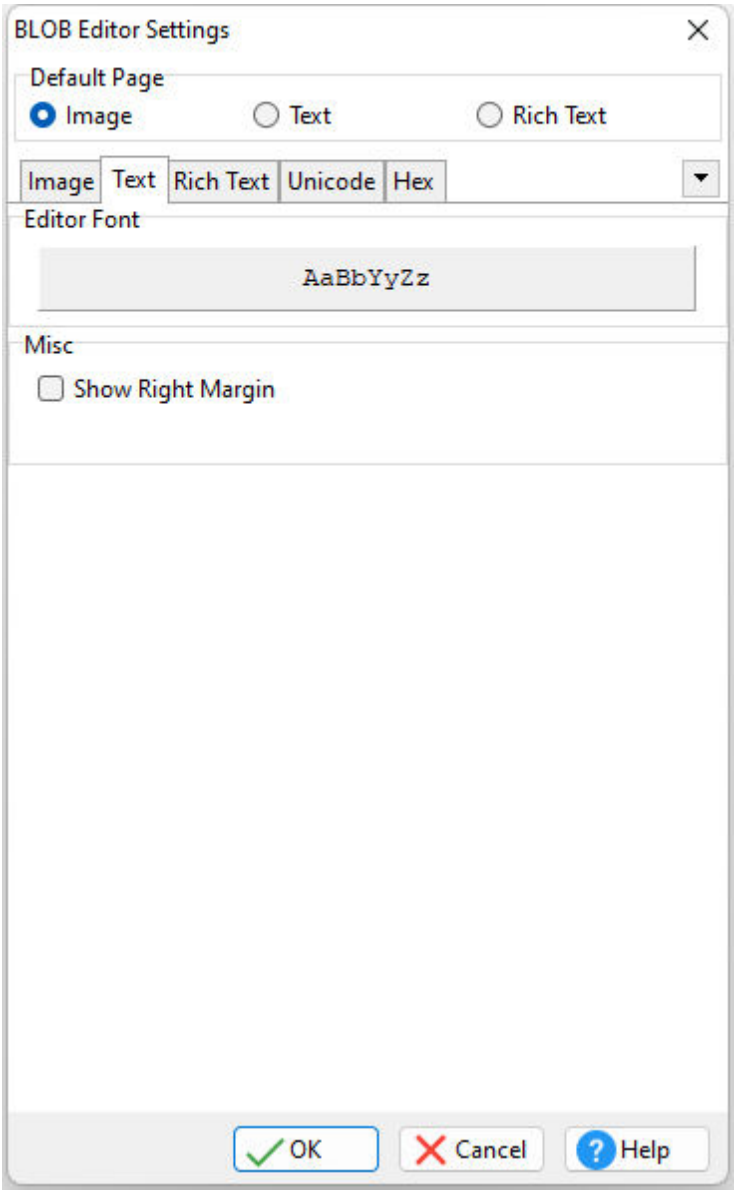
2.10.2 Text

⇨ **Default Page**
Specifies the default page displayed when the BLOB Editor is launched for an empty data set. If data exists for the column, the BLOB Editor will default to the appropriate page based on the type of data stored.

⇨ **Editor Font**
Specifies the editor text font type, size, and style

⇨ **Misc**
Specifies if the right margin is displayed

Copyright © 1982-2024 R:BASE Technologies, Inc.



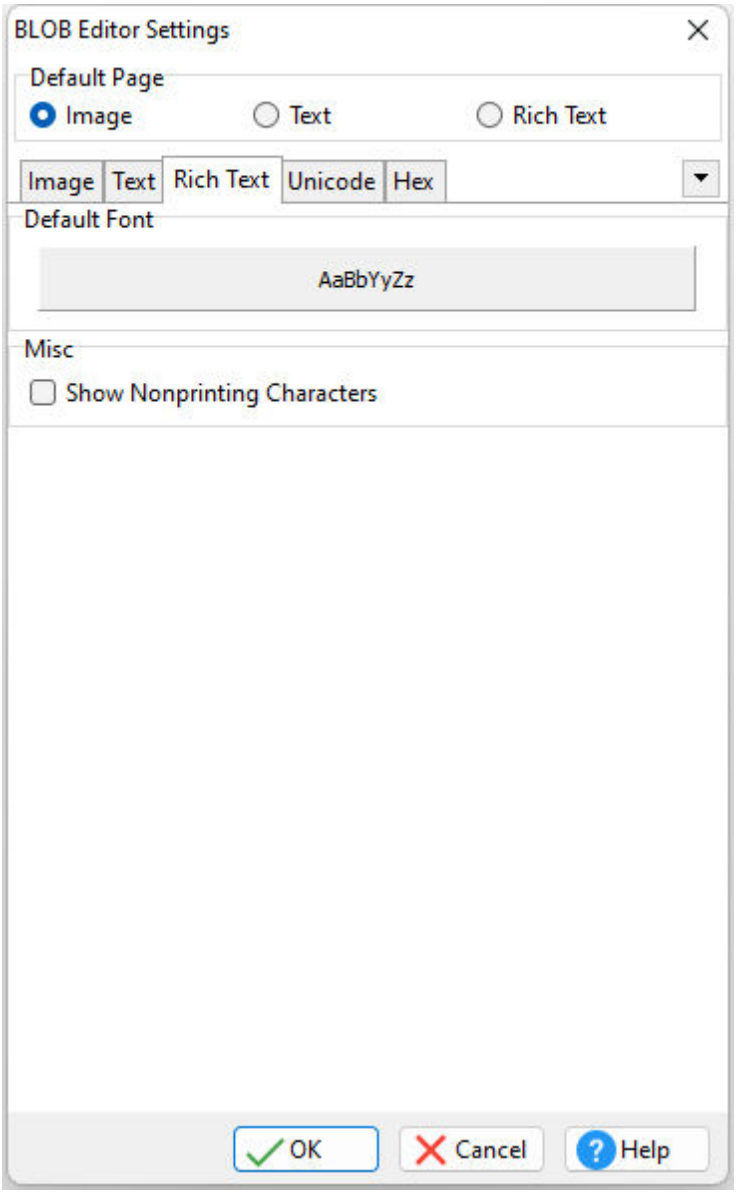
2.10.3 Rich Text

⇨ **Default Page**
Specifies the default page displayed when the BLOB Editor is launched for an empty data set. If data exists for the column, the BLOB Editor will default to the appropriate page based on the type of data stored.

⇨ **Default Font**
Specifies the default text font type, size, and style

⇨ **Misc**
Specifies if nonprinting characters are displayed

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.10.4 Unicode

⇨ **Default Page**
Specifies the default page displayed when the BLOB Editor is launched for an empty data set. If data exists for the column, the BLOB Editor will default to the appropriate page based on the type of data stored.

⇨ **Editor Font**
Specifies the editor font type, size, and style

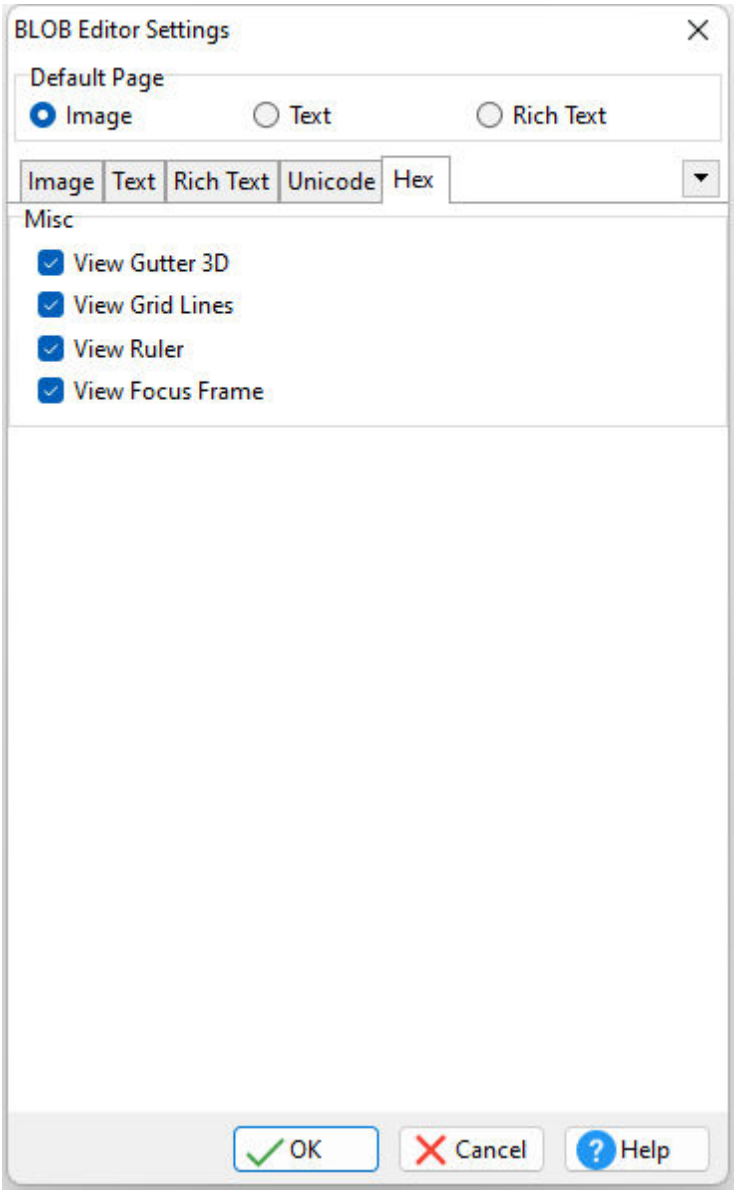
Copyright © 1982-2024 R:BASE Technologies, Inc.

2.10.5 Hex

⇨ **Default Page**
Specifies the default page displayed when the BLOB Editor is launched for an empty data set. If data exists for the column, the BLOB Editor will default to the appropriate page based on the type of data stored.

⇨ **Misc**
View Gutter 3D - specifies if the gutter lines are displayed
View Grid Lines - specifies if the grid lines are displayed
View Ruler - specifies if the ruler is displayed
View Focus Frame - specifies if a selection displays a solid frame or the dotted focus frame

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.11 Hint Settings...

The Hint Settings selection allows you to change the appearance of all R:BASE program hints. Your changes will alter the hints used by the R:BASE development environment as well as the hints specified in your application, forms, etc.

⇨ **Hint Type**
Specifies the hint type

⇨ **Background Color and Font** Specifies the background color and text font, color and style. The Background Color only effects "Regular" and "Balloon" hints.

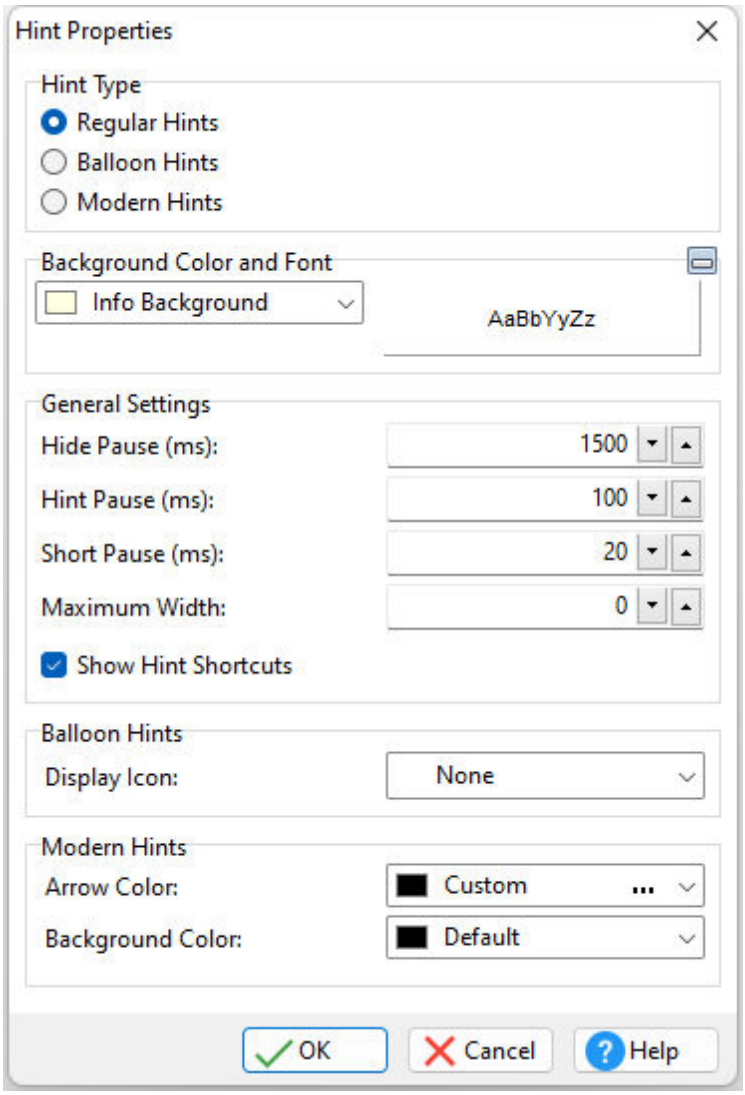
⇨ **General Settings**
Hide Pause (ms) - specifies the time interval for the hint to be displayed on the screen
Hint Pause (ms) - specifies the time interval for the hint to appear after the cursor first moves over the control
Short Pause (ms) - specifies the time interval for the hint to appear after the cursor leaves and immediately returns to the same control.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Maximum Width - specifies the maximum width for the hint message
Show Hint Shortcuts - hint shortcuts will be displayed

⇨ **Balloon Hints**
Display Icon - specifies the icon that can be displayed within balloon hints

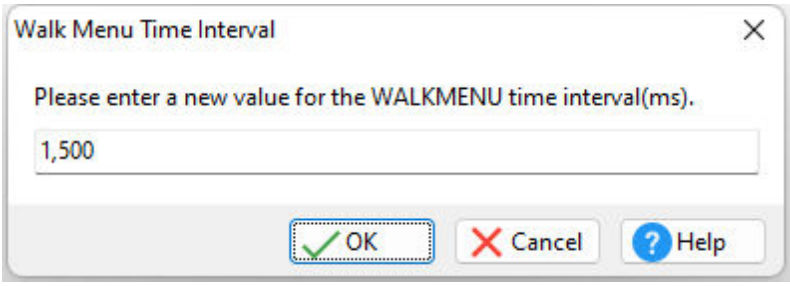
⇨ **Modern Hints**
Arrow Color - specifies the color of the modern hint's arrow in the thick border
Background Color - specifies the color of the modern hint's text background



Copyright © 1982-2024 R:BASE Technologies, Inc.

2.12 Walkmenu Time Interval...

This selection allows you to set the WALKMENU time interval for how often keystrokes are recognized.

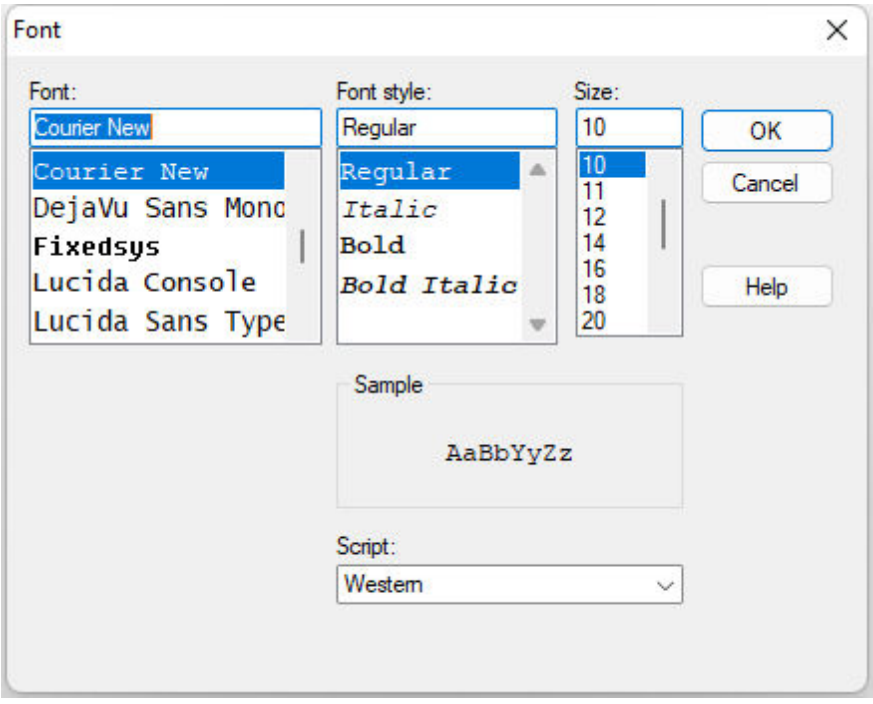


WALKMENU is a menu shortcut function allowing the user to access menu selections by typing the beginning characters (up to when a match is made) of their names. Pressing any navigational keys (such as [Home] or [Page Up]) clears the buffer containing the keystrokes entered by the user while traversing the menu list. Any keystrokes not resulting in a match are not stored in the buffer, causing a beep.

2.13 Default Printer Font...

Specifies the default printer font for output generated with the OUTPUT PRINTER command. The default printer font is also recognized with the "OUTPUT filespec PDF" command when the OUTPUT_PDF_USE_PRINTER_FONT PROPERTY command parameter is set ON:

```
PROPERTY APPLICATION OUTPUT_PDF_USE_PRINTER_FONT ON
```

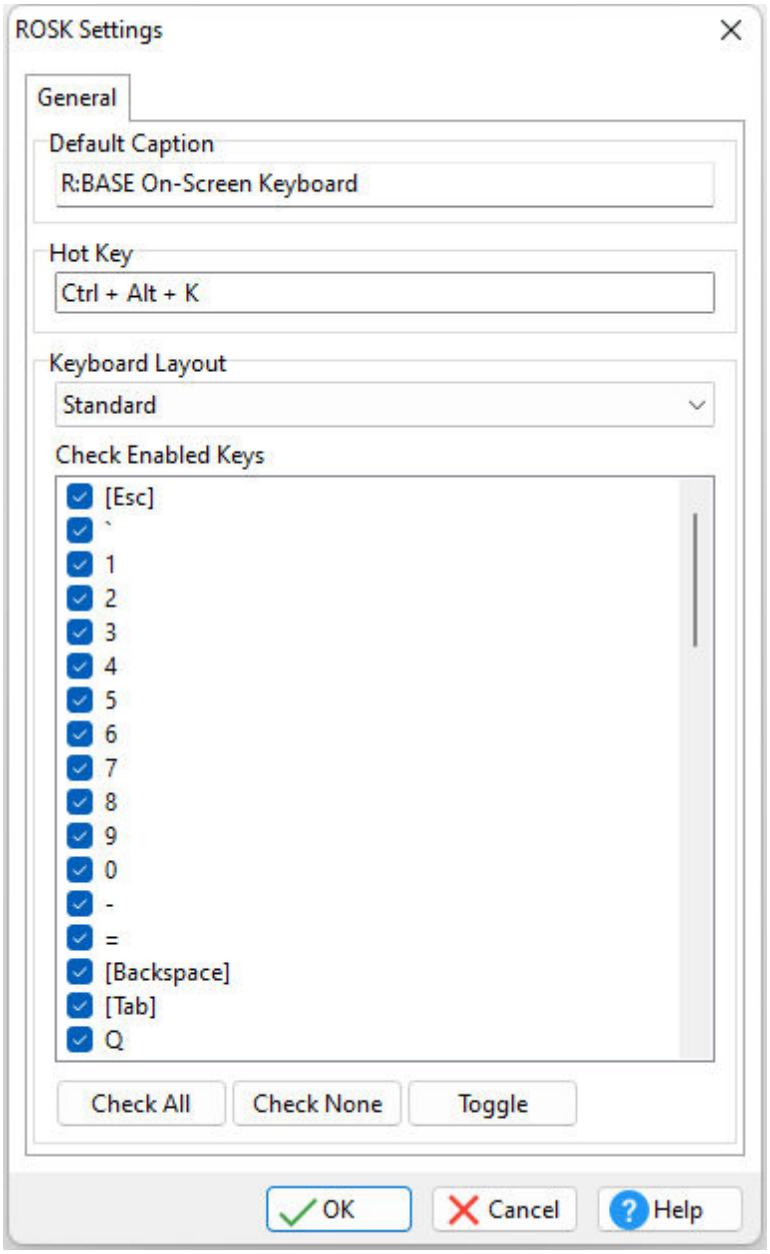


Copyright © 1982-2024 R:BASE Technologies, Inc.

2.14 ROSK Settings...

Provides settings for the R:BASE On-Screen Keyboard (ROSK)

- *Default Caption* - specifies the default window caption when the keyboard is launched
- *Hot Key* - specifies the hot key to display/hide the keyboard
- *Keyboard Layout* - specifies the layout, Standard or NumPad
- *Check Enable Keys* - specifies the keys to display

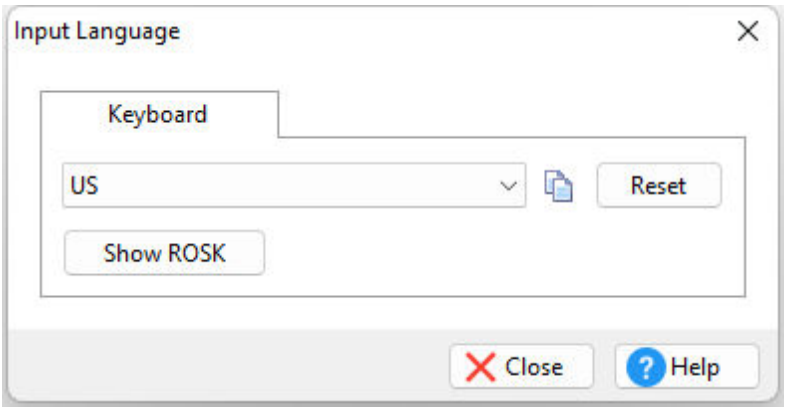


2.15 Input Language...

The Input Language setting alters the keyboard input language within R:BASE. Keyboard types for languages and custom setups are available. Languages can also be specified using a language name or an 8-digit input locale with the PROPERTY command.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Show ROSK - displays the R:BASE On-Screen Keyboard to preview the selected input language. When the input language is changed, the ROSK display will change and input will appear as listed on the keys.
Copy to Clipboard - captures the current keyboard selection and builds the PROPERTY command syntax to utilize the selected keyboard
Reset - resets the language



Notes:

- When the input language keyboard for program processing is changed, so is the physical keyboard. What you see on the ROSK is also the keys implemented upon the physical keyboard.
- In the case where the user alters the input language and cannot go back to the default settings using a command, the input language can reset by selecting "Settings" > "Input Language" from the menu bar, and clicking on the "Reset" button.
- In Windows versions before 8.1, every process/program can have its own input language setting (e.g. Notepad language is Russian, Word in Dutch, the rest in English). In Windows 8.1 and above, the input language setting of one process/program will change the input language setting for the entire operating system.

PROPERTY Command
A new PROPERTY command parameter has been introduced for the ability to set the input language on demand, where the below *Value* specifies a predefined name or an 8-digit input locale.

```
PROPERTY APPLICATION INPUT_LANGUAGE Value
```

Predefined names include: Arabic, Default, Dutch, English, French, German, Italian, Portuguese, Spanish, Russian, and Ukrainian.

Examples
Example 01: To use a named language for Russian:

```
PROPERTY APPLICATION INPUT_LANGUAGE RUSSIAN
```

Example 02: To use the 8-character locale for Albanian:

```
PROPERTY APPLICATION INPUT_LANGUAGE 0000041C
```

Example 03: To reset the Input Language:

```
PROPERTY APPLICATION INPUT_LANGUAGE DEFAULT
```

Additional Languages
Other languages can be specified for the keyboard input by specifying an 8-digit input locale.

Locale	Input Locale	Language Collection
Afrikaans	00000409	Basic

Copyright © 1982-2024 R:BASE Technologies, Inc.

Albanian	0000041c 00000409	Basic
Arabic_Saudi_Arabia, Arabic_Iraq, Arabic_Egypt, Arabic_Oman, Arabic_Yemen, Arabic_Syria, Arabic_Jordan, Arabic_Lebanon, Arabic_Kuwait, Arabic_UAE, Arabic_Bahrain	00000409 00000401	Complex Script
Arabic_Libya, Arabic_Algeria, Arabic_Morocco, Arabic_Tunisia	0000040c 00020401	Complex Script
Arabic_Qatar	00000409 00000401	Complex Script
Armenian	0000042b 00000409 00000419	Complex Script
Azeri_Latin	0000042c 0000082c 00000419	Basic
Azeri_Cyrillic	0000082c 0000042c 00000419	Basic
Basque	0000040a 00000409	Basic
Belarusian	00000423 00000409 00000419	Basic
Bengali_India	00000445 00000409	Complex Script
Bosnian_Latin	0000141A 00000409	Basic
Bulgarian	00000402 00000409	Basic
Catalan	0000040a 00000409	Basic
Chinese_Taiwan	00000404 e0080404 E0010404	East Asian
Chinese_PRC	00000804 e00e0804 e0010804 e0030804 e0040804	East Asian
Chinese_Hong_Kong	00000409 e0080404	East Asian
Chinese_Singapore	00000409 e00e0804 e0010804 e0030804 e0040804	East Asian
Chinese_Macau	00000409 e00e0804 e0020404 e0080404	East Asian
Croatian	0000041a 00000409	Basic
Croatian_Bosnia_Herzegovina	0000041a 00000409	Basic
Czech	00000405 00000409	Basic
Danish	00000406 00000409	Basic
Divehi	00000409 00000465	Complex Script
Dutch_Standard	00020409 00000413	Basic

Copyright © 1982-2024 R:BASE Technologies, Inc.

	00000409	
Dutch_Belgian	00000813 00000409	Basic
English_United_States, English_Australian, English_New_Zealand, English_South_Africa, English_Jamaica, English_Caribbean, English_Belize, English_Trinidad, English_Zimbabwe, English_Philippines	00000409	Basic
English_United_Kingdom	00000809	Basic
English_Canadian	00000409 00011009 00001009	Basic
English_Ireland	00001809 00011809	Basic
Estonian	00000425	Basic
Faeroese	00000406 00000409	Basic
Farsi	00000409 00000429 00000401	Complex Script
Finnish	0000040b 00000409	Basic
French_Standard, French_Luxembourg, French_Monaco	0000040c 00000409	Basic
French_Belgian	0000080c 00000409	Basic
French_Canadian	00011009 00000409	Basic
French_Swiss	0000100c 00000409	Basic
Georgian	00000437 00000409 00000419	Complex Script
Galician	0000040a 00000409	Basic
German_Standard, German_Austrian, German_Luxembourg, German_Liechtenstein	00000407 00000409	Basic
German_Swiss	00000807 00000409	Basic
Greek	00000408 00000409	Basic
Gujarati	00000409 00000447 00010439	Complex Script
Hebrew	00000409 0000040d	Complex Script
Hindi	00000409 00010439 00000439	Complex Script
Hungarian	0000040e 00000409	Basic
Icelandic	0000040f 00000409	Basic
Indonesian	00000409	Basic
Italian_Standard, Italian_Swiss	00000410 00000409	Basic
Japanese	e0010411	East Asian
Kannada	00000409 0000044b 00010439	Complex Script
Kazakh	0000043f 00000409 00000419	Basic

Copyright © 1982-2024 R:BASE Technologies, Inc.

Konkani	00000409 00000439	Complex Script
Korean	E0010412	East Asian
Kyrgyz	00000440 00000409	Basic
Latvian	00010426	Basic
Lithuanian	00010427	Basic
Macedonian	0000042f 00000409	Basic
Malay_Malaysia, Malay_Brunei_Darussalam	00000409	Basic
Malayalam	00000409 0000044c	Complex Script
Maltese	00000409 0000043a	Basic
Maori	00000409 00000481	Basic
Marathi	00000409 0000044e 00000439	Complex Script
Mongolian	00000450 00000409	Basic
Norwegian_Bokmal, Norwegian_Nynorsk	00000414 00000409	Basic
Polish	00010415 00000415 00000409	Basic
Portuguese_Brazilian	00000416 00000409	Basic
Portuguese_Standard	00000816 00000409	Basic
Punjabi	00000409 00000446 00010439	Complex Script
Quechua_Bolivia, Quechua_Ecuador, Quechua_Peru	00000409 0000080A	Basic
Romanian	00000418 00000409	Basic
Russian	00000419 00000409	Basic
Sami_Inari	0001083b 00000409	Basic
Sami_Lule_Norway, Sami_Northern_Norway, Sami_Southern_Norway	0000043b 00000409	Basic
Sami_Lule_Sweden, Sami_Northern_Sweden, Sami_Southern_Sweden	0000083b 00000409	Basic
Sami_Northern_Finland, Sami_Skolt	0001083b 00000409	Basic
Sanskrit	00000409 00000439	Complex Script
Serbian_Latin	0000081a 00000409	Basic
Serbian_Latin_Bosnia_Herzegovina	0000081a 00000409	Basic
Serbian_Cyrillic	00000c1a 00000409	Basic
Serbian_Cyrillic_Bosnia_Herzegovina	00000c1a 00000409	Basic
Slovak	0000041b 00000409	Basic
Slovenian	00000424 00000409	Basic
Spanish_Traditional_Sort, Spanish_Modern_Sort	0000040a 00000409	Basic

Copyright © 1982-2024 R:BASE Technologies, Inc.

Spanish_Mexican, Spanish_Guatemala, Spanish_Panama, Spanish_Dominican_Republic, Spanish_Venezuela, Spanish_Colombia, Spanish_Peru, Spanish_Argentina, Spanish_Ecuador, Spanish_Chile, Spanish_Uruguay, Spanish_Paraguay, Spanish_Bolivia, Spanish_El_Salvador, Spanish_Honduras, Spanish_Nicaragua, Spanish_Puerto_Rico	0000080a 00000409	Basic
Swahili	00000409	Basic
Swedish, Swedish_Finland	0000041d 00000409	Basic
Syriac	00000409 0000045a	Complex Script
Tamil	00000409 00000449	Complex Script
Tatar	00000444 00000409 00000419	Basic
Telugu	00000409 0000044a 00010439	Complex Script
Thai	00000409 0000041e	Complex Script
Tswana	00000409 0000041f	Basic
Ukrainian	00000422 00000409	Basic
Turkish	0000041f	Basic
Ukrainian	00000422 00000409	Basic
Urdu	00000401 00000409	Complex Script
Uzbek_Latin	00000409 00000843 00000419	Basic
Uzbek_Cyrillic	00000843 00000409 00000419	Basic
Vietnamese	00000409 0000042a	Complex Script
Welsh	00000452 00000809	Basic
Xhosa	00000409	Basic
Zulu	00000409	Basic

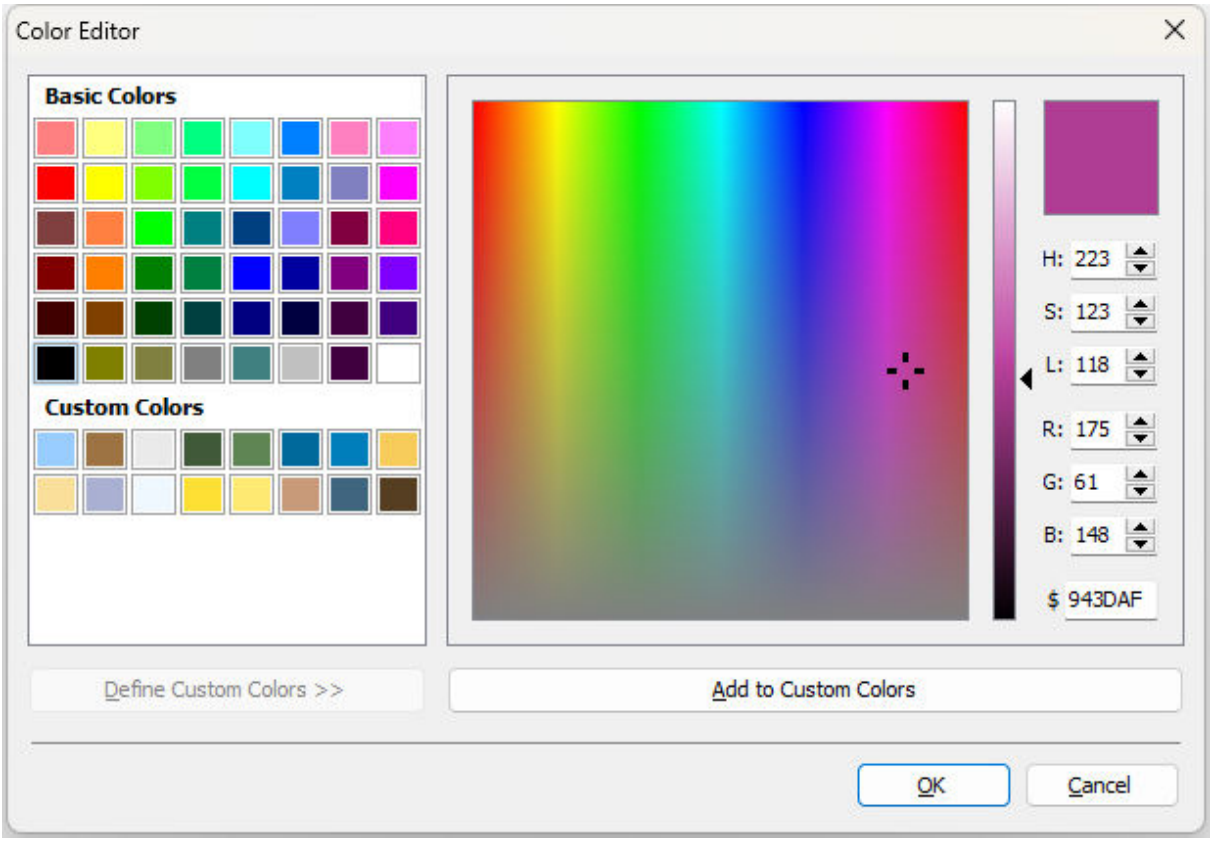
2.16 Custom Colors

Allows users to save and load custom colors from one development computer to another.

Users may also switch between Delphi or HTML RGB Hex notation. The default is Delphi notation. Most colors (online color combinations) are presented in HTML format (#AARRGGBB), where being able to accept that format directly from a copy/paste source can save time in active development.

In Delphi mode, the Color Editor will appear as "$ [.....]" in the bottom right corner. In HTML mode, the Color Editor will appear as "# [.....]".

Copyright © 1982-2024 R:BASE Technologies, Inc.

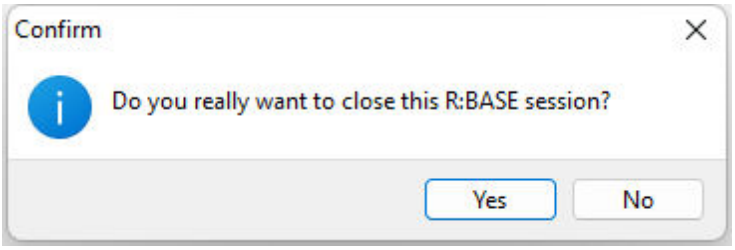


2.17 Registry Settings

Allows users to save and load custom R:BASE environment settings from one development computer to another.

2.18 Warn When Closing R:BASE session

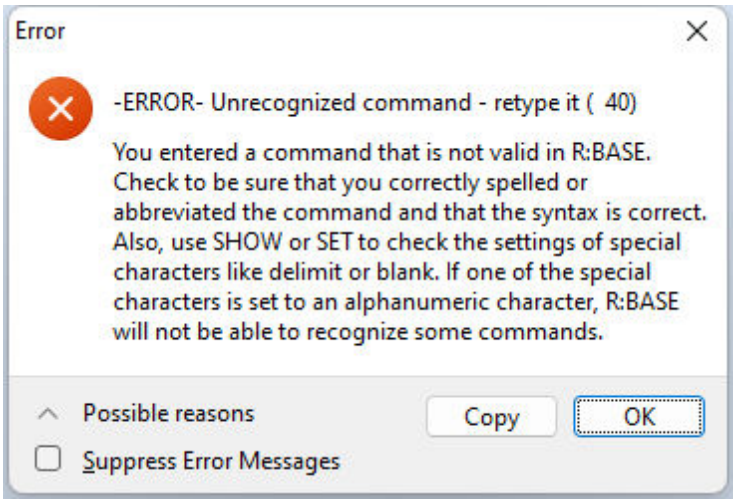
This selection toggles the display of a warning when closing the R:BASE session.



2.19 Show Check Box to Suppress Error Messages

This selection places a check box in all error message windows allowing you to suppress the error message(s). After selecting to suppress the error messages, you will not see any error messages within the entire R:BASE environment. In order for error messages to be displayed again, you must restart R:BASE or issue the "SET ERROR MESSAGES ON" command.

Copyright © 1982-2024 R:BASE Technologies, Inc.



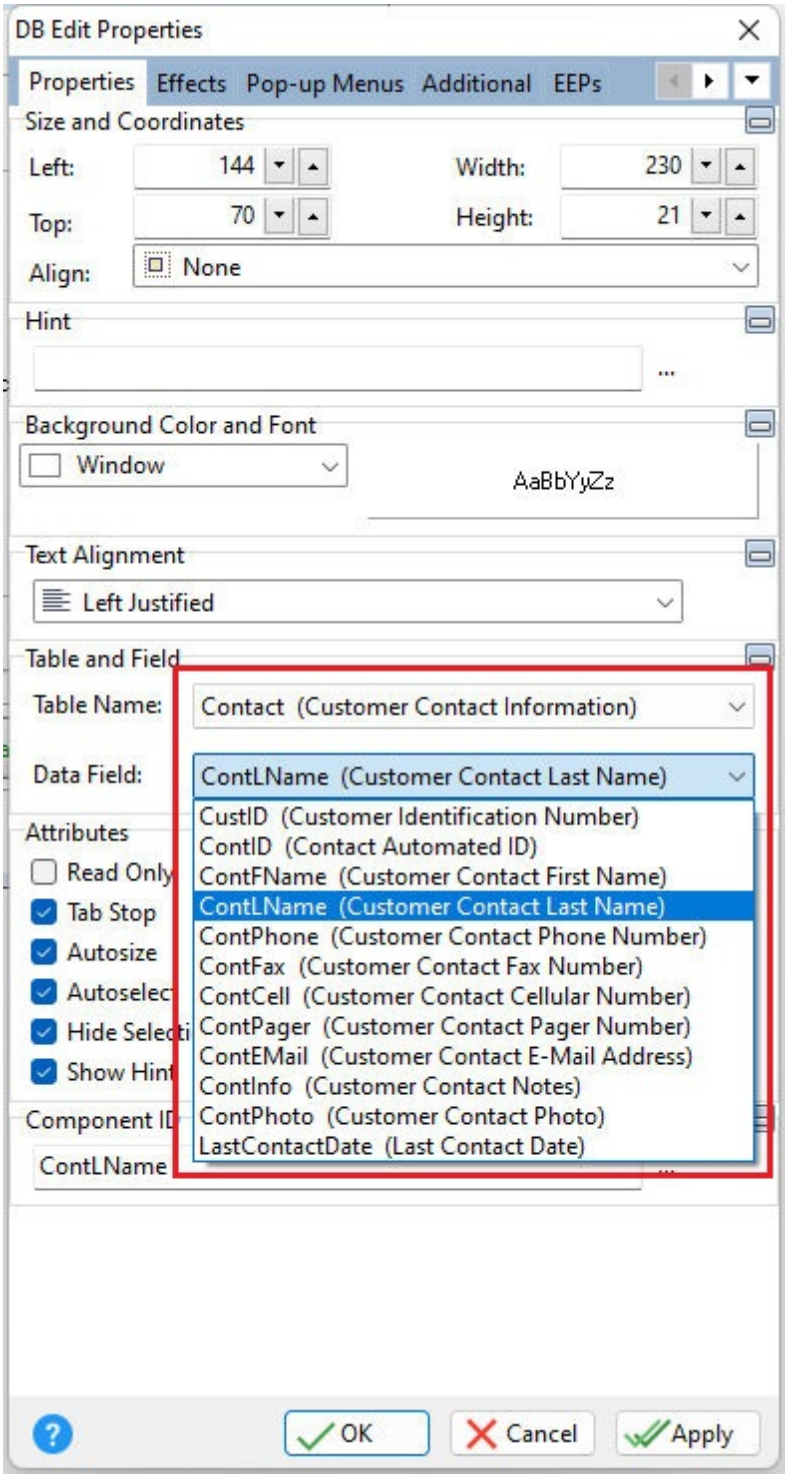
2.20 Clean Scratch Files on Exit

Enabling this option deletes any remaining temporary scratch files (.$$$) after R:BASE closes.

2.21 Show Comments in Table/Field List

The selection toggles the display of comments/descriptions within property editors and table listing dialogs.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.22 Configuration Settings...

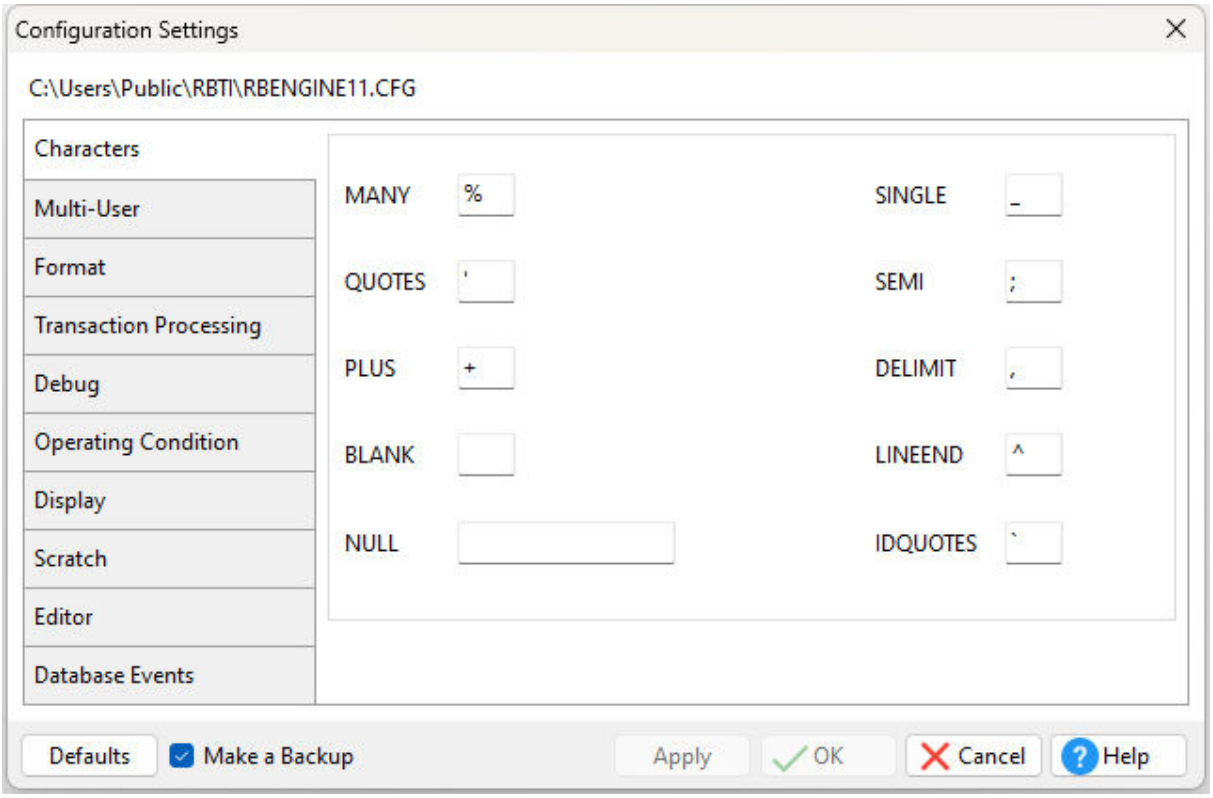
This selection allows you to alter the settings that are stored in the R:BASE configuration file. The path for the R:BASE configuration file in use (RBENGINE11.CFG) is displayed across the top of the Configuration Settings dialog.

Copyright © 1982-2024 R:BASE Technologies, Inc.

2.22.1 Characters

- *MANY* - specifies the character used to specify a pattern for more than one character. The default is the percent (%).
- *QUOTES* - specifies the character used when specifying character strings. The default is an apostrophe(').
- *PLUS* - specifies the character used to indicate that a line wraps in code. The default is the plus (+).
- *BLANK* - specifies the character used to indicate a space. The default is a space.
- *NULL* - specifies up to four characters to indicate a null value. The default is a hyphen, zero, and hyphen (-0-).
- *SINGLE* - specifies the character used to specify a pattern for one character. The default is an underscore (_).
- *SEMI* - specifies the character used to separate commands. The default is the semi-colon (;).
- *DELIMIT* - specifies the character used to separate characters or strings. The default is the comma (,).
- *LINEEND* - specifies the character used to indicate the end of a line for display of fields with NOTE or TEXT data types, or in the Data Browser. The default character for R:BASE database that migrated over the years from DOS versions is þ, the ASCII value 0254. The default character for R:BASE for Windows is the carat (^).
- *IDQUOTES* - specifies the character used for enclosing object names, such as column or table names. The default is the reverse quote (`).

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.



Copyright © 1982-2024 R:BASE Technologies, Inc.

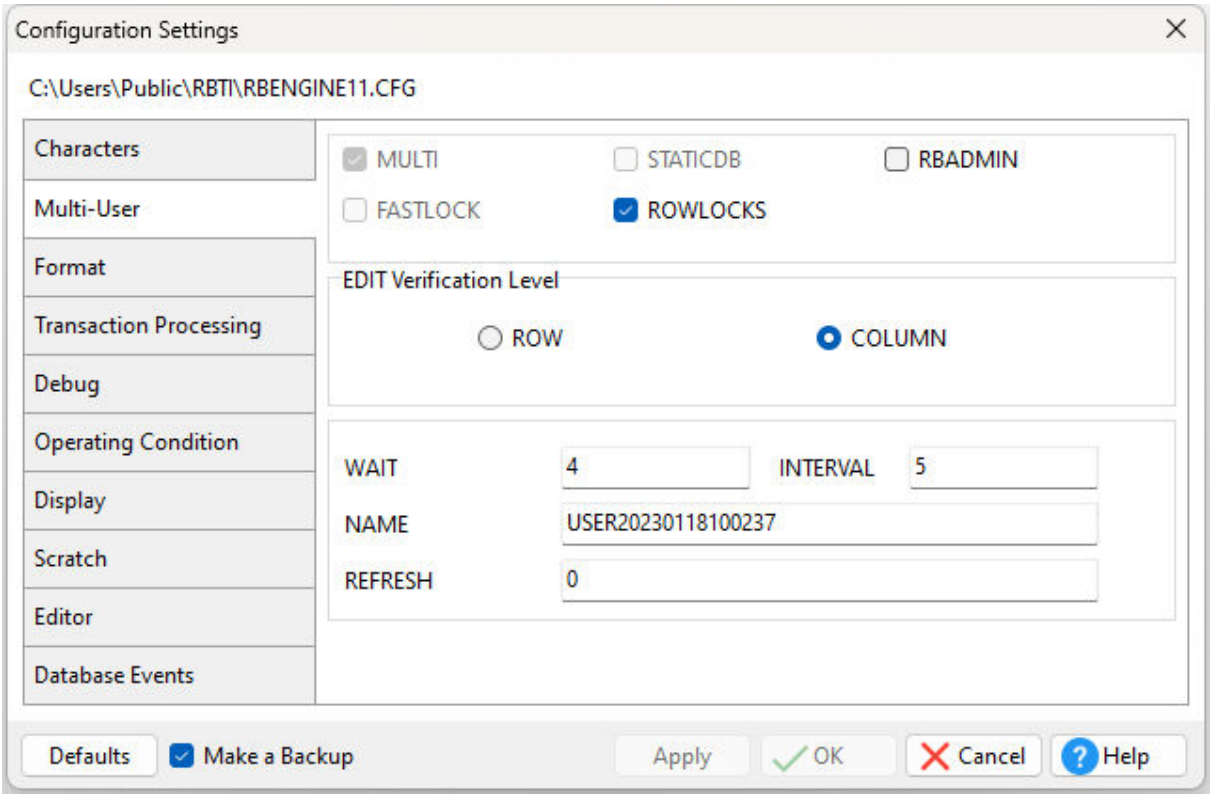
2.22.2 Multi-User

- *MULTI* - toggles R:BASE use in a multi-user environment
- *FASTLOCK* - provides faster multi-user performance while modifying data. With FASTLOCK on, R:BASE does not place a table lock on the table, allowing for greater throughput. A table lock is only needed to prevent structure changes.
- *STATICDB* - prevents users from making permanent structure changes in the database. Any tables created will be temporary and will be discarded when they exit R:BASE. See also STATICDB.
- *ROWLOCKS* - toggles row-locking. When this option is not checked, while one user is updating a table, R:BASE will not allow other users to access any row in the table. If you check this option, R:BASE only denies access to the specific row being updated, not the entire table.
- *RBADMIN* - toggles the support for RBAdmin in a network environment

- *EDIT* Verification Level - sets the concurrency control. column in that row. sets the concurrency control. When set to **ROW** and a user modifies a row of data in a table, R:BASE checks if other users are modifying data in any column in that row. If R:BASE notes a change, the entire form is refreshed for the user with the data as it currently exists in the database. When set to **COLUMN** and a user modifies data in a column in a table, R:BASE checks if other users are modifying data in that column only. As long as there is no conflict, other users' changes appear without a warning message, and the user's edited data is saved to its table.

- *WAIT* - the time in seconds for which R:BASE will continue to retry access to a table or database before it halts.
- *INTERVAL* - the time in tenths of a second that R:BASE wakes up to try again to access a table or database.
- *NAME* - the name used to distinguish this workstation from others.
- *REFRESH* - the interval specifying how often R:BASE refreshes a form and displays current data.

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.



Copyright © 1982-2024 R:BASE Technologies, Inc.

2.22.3 Format

⇨ **Date**
- *FORMAT* - the format for the way you want the date displayed when you view database information. For example, if you want reports to display dates with hyphens, enter "MM-DD-YY."
- *YEAR* - specifies the century threshold year
- *SEQUENCE* - the format for the way you want dates entered. For example, if you want to be able to enter dates in forms in international style, enter "MMDDYY."
- *CENTURY* - specifies the default century (the first two digits of a four-digit year)

⇨ **Time**
- *FORMAT* - the format for the way you want the time displayed when you view database information. For example, if you want forms to display the time in hours and minutes only, enter "HH:MM."
- *SEQUENCE* - the format for the way you want the time entered. For example, if you want to be able to enter the time in forms as hours then minutes, enter "HHMM."

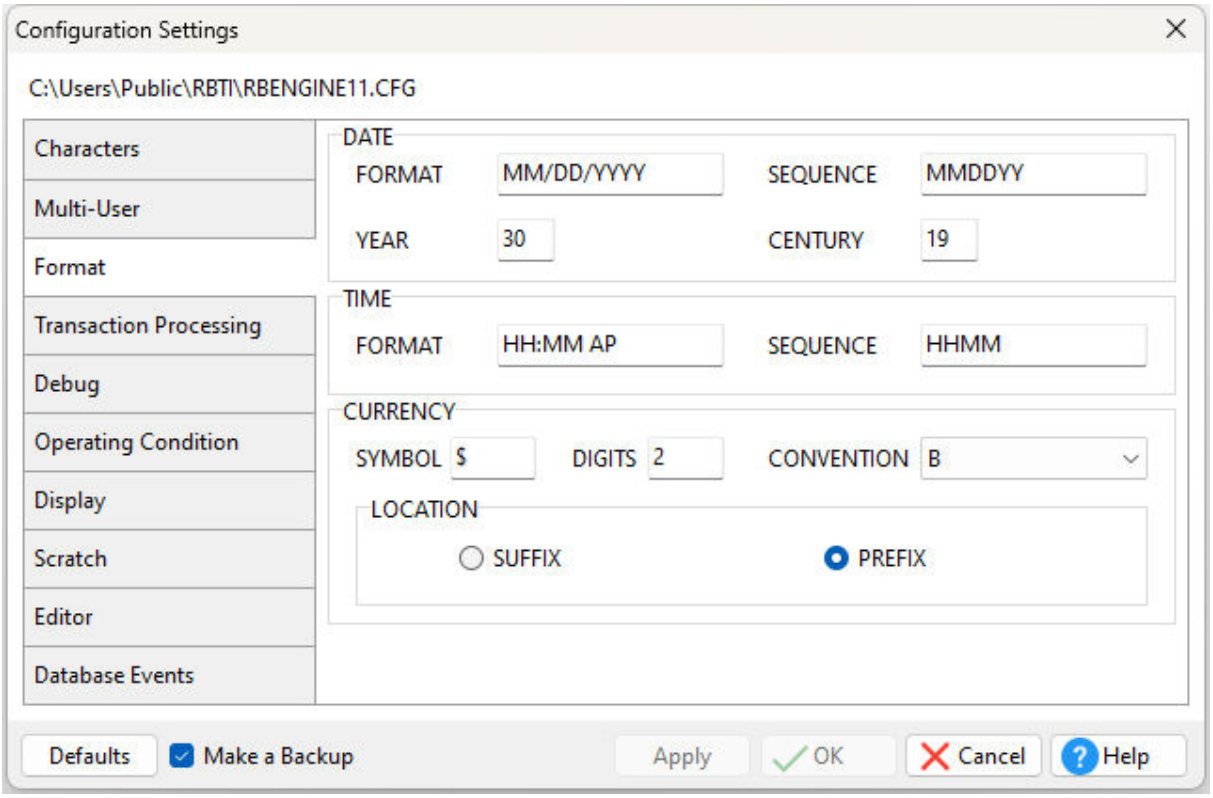
⇨ **Currency**
- *SYMBOL* - specifies up to four characters for the currency symbol
- *DIGITS* - specifies the number of digits (up to 16) to indicate the number of decimal places to display for currency. The default is two and displays two decimal places for cents.
- *CONVENTION* - indicates which format to use for decimal and thousands delimiters. You can select the following formats:

Convention	Thousands Delimiter	Decimal Delimiter	Example
A	.	,	123.456.793,01
B	,	.	123,456,793.01
C	(blank)	,	123 456 793,01
D	N/A	.	123456793.01
E	'	.	123'456'793.01

⇨ **Location**
- *SUFFIX* - specifies if you want the currency symbol to follow the currency value, such as "120 FF."
- *PREFIX* - specifies if you want the currency symbol to display before the currency value, such as "$120.00."

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.

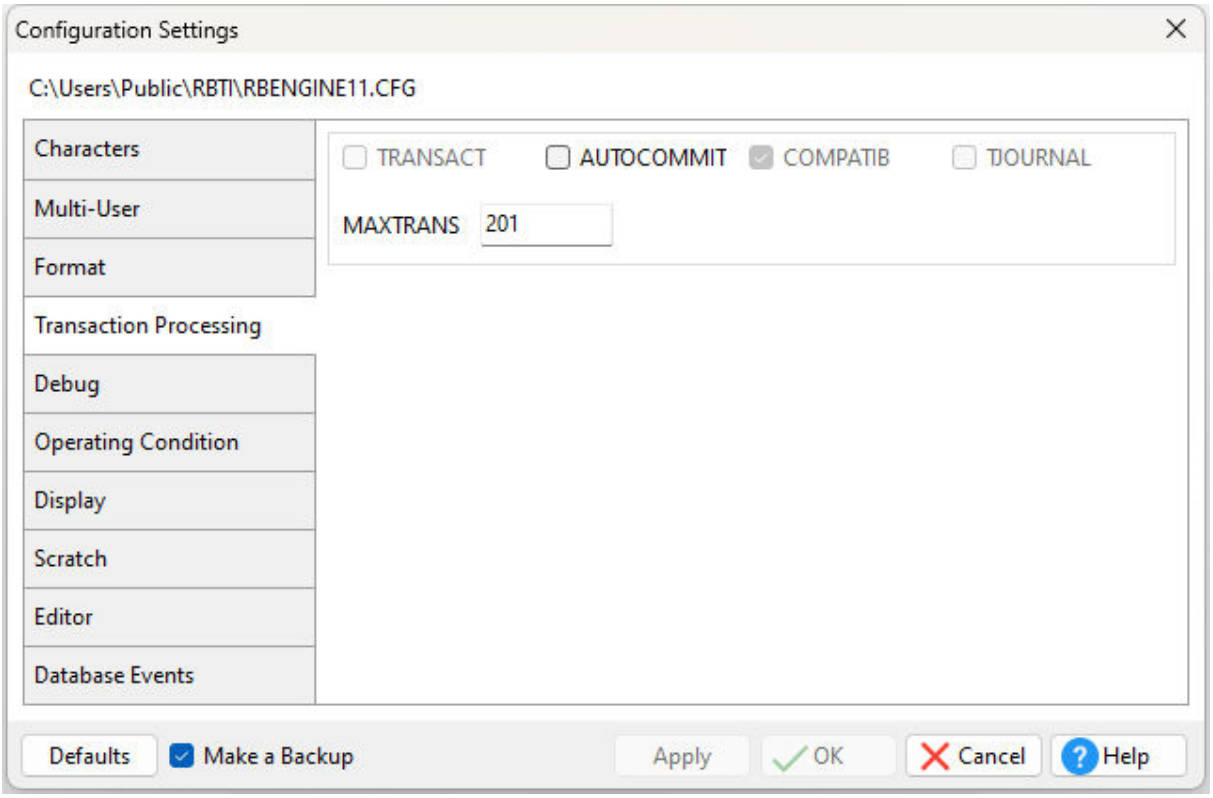


2.22.4 Transaction Processing

- *TRANSACT* - toggles transaction processing on and off. When transaction processing is set on and AUTOCOMMIT is set off, all commands entered after one COMMIT or ROLLBACK command until the next comprise a transaction.
- *AUTOCOMMIT* - toggles AUTOCOMMIT processing on and off. When transaction processing and AUTOCOMMIT are on, each command that is executed successfully is immediately made permanent and visible to network users.
- *COMPATIB* - toggles compatibility with R:BASE transactions with prior R:BASE versions and Oterro.
- *TJOURNAL* - toggles journaling of commands that modify data when transaction processing is on.
- *MAXTRANS* - specifies the maximum number of users who can have the same database open concurrently with transaction processing on.

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.

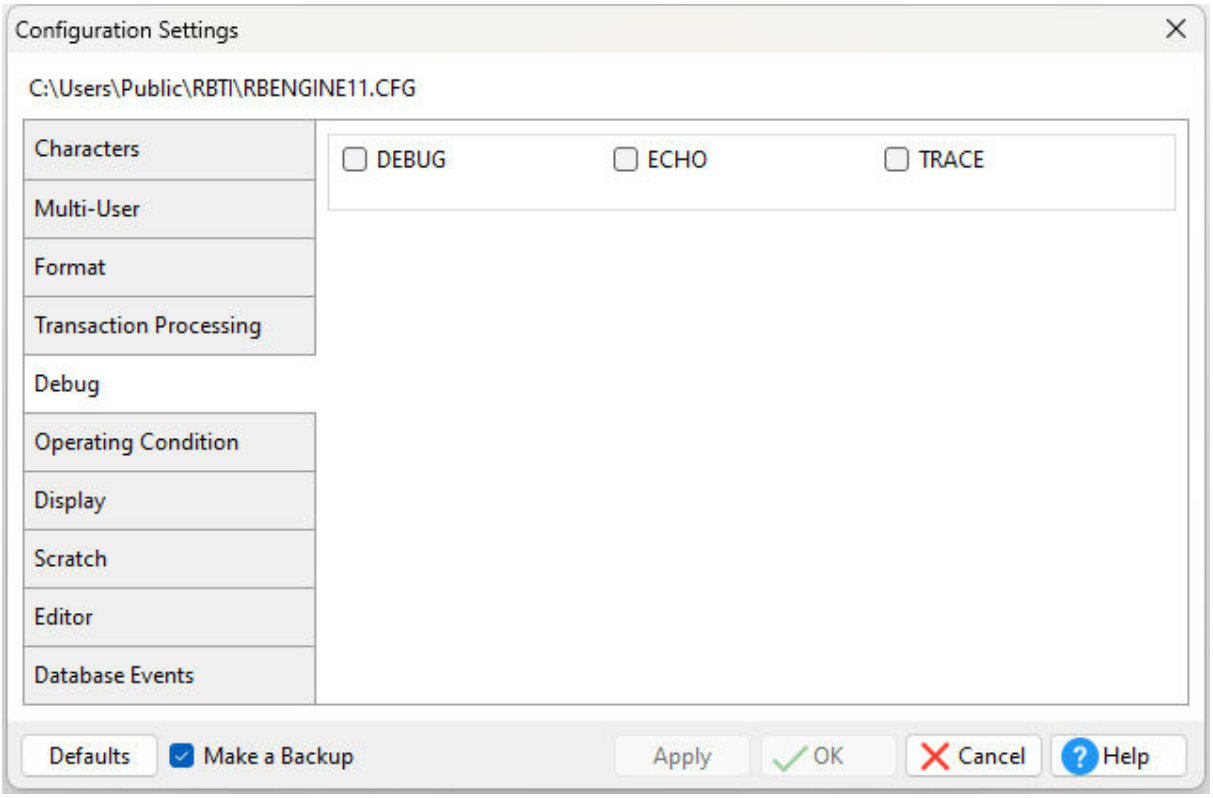


2.22.5 Debug

- *DEBUG* - allows R:BASE to execute commands that are preceded with "DEBUG." If this option is not checked, R:BASE ignores these commands.
- *ECHO* - displays commands as they are processed from the current ASCII input device
- *TRACE* - execute TRACE (Interactive Command File Debugger) inside a command file to trace a block of code as defined

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.22.6 Operating Condition

- *AND* - gives the AND operator precedence over the OR operator. R:BASE processes all AND before all OR conditions, and all AND NOT before AND OR conditions. If this option is not checked, R:BASE processes conditions from left to right.
- *ANSI* - reduces the reserved word list. For this list, see "Reserved Words".
- *ANSIOUTPUT* - specifies ANSI is the default encoding method for the OUTPUT command
- *AUTOCONVERT* - automatically converts R:BASE databases created in prior versions
- *AUTODROP* - controls the feature for a combo-box in a form to automatically drop-down the list when it gets focus
- *AUTORECOVER* - errors that can occur during transaction processing are automatically corrected when the program is interrupted, for example from a network or power failure
- *AUTOROWVER* - used for Oterro compatibility only. If AUTOROWVER is set on, every CREATE TABLE or ALTER TABLE command will add the SYS_ROWVER column if it does not already exist
- *AUTOSKIP* - allows Autoskip in forms; after entering a value in a field, the cursor automatically moves to the next field. If this option is not checked, you must press [Tab] to move to the next field.
- *AUTOSYNC* - connecting to a database will automatically synchronize the database files if necessary
- *AUTOUPGRADE* - converts R:BASE 6.0 databases to the current release and adds the new system tables for handling stored procedures and triggers
- *BELL* - sounds the bell when an error occurs
- *BOOLEAN* - constants (e.g. TRUE, FALSE) in expressions will be treated as type BOOLEAN values
- *CASE* - makes a distinction between lowercase and uppercase in WHERE clauses, IF structures, WHILE loops, TALLY, and in RULES where are comparisons are equal or not equal.
- *CHECKPROP* - displays PROPERTY/GETPROPERTY ERROR messages during processing
- *CLEAR* - allows R:BASE to clear the internal buffers and transfer data to disk after each modification. If this option is not checked, R:BASE writes modified data to disk only when the buffer is full, a database is closed, or you exit R:BASE.
- CMDHIST - Specifies to update the R> Prompt Command History when running commands from dialogs
- *CMPAUSE* - (Cascade Modal Pause) determines if R:BASE will use a local PAUSE dialog window for modal PAUSE displays instead of the global PAUSE form
- *COLCHECK* - columns will be checked for consistency when making views

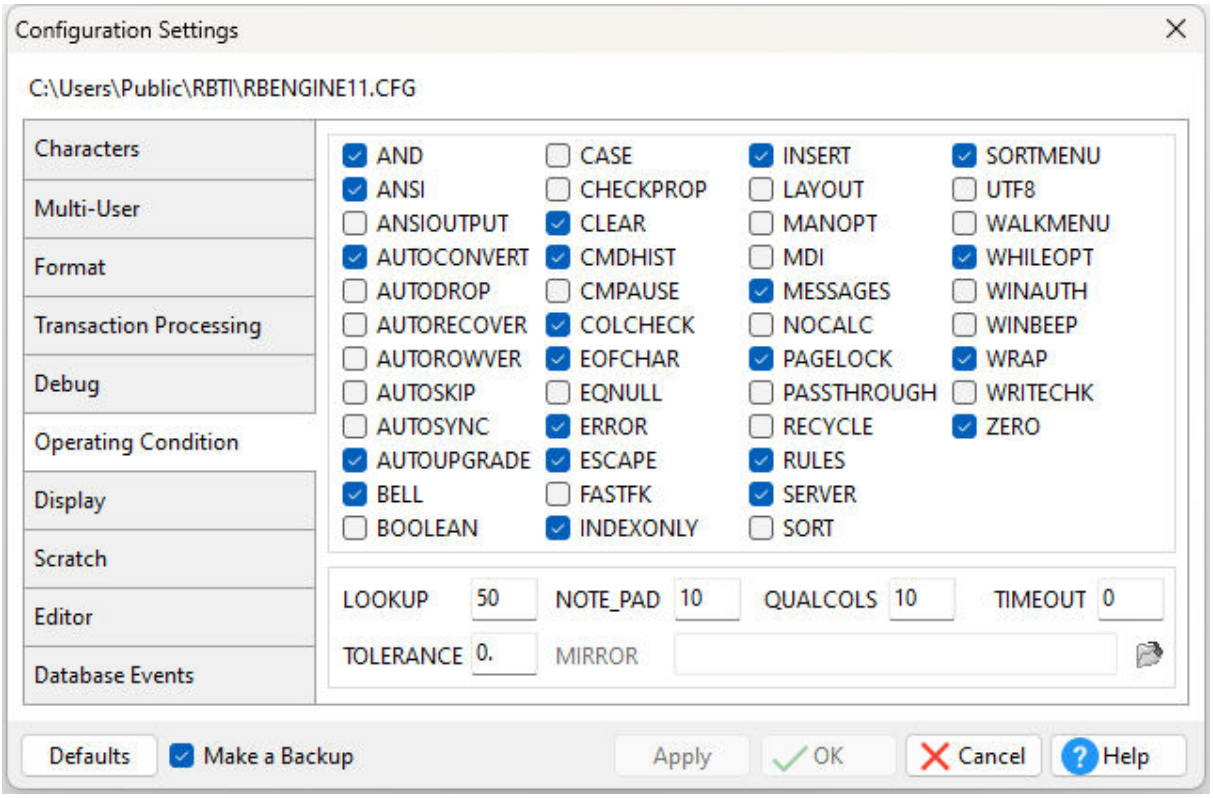
Copyright © 1982-2024 R:BASE Technologies, Inc.

- *EOFCHAR* - controls whether a control-Z character will appended to the end of output files
- *EQNULL* - determines how comparison of a NULL variable and a non-NULL constant are calculated
- *ERROR* - toggles the display of error messages
- *ESCAPE* - allows the ability to stop processing in the middle of command files, WHILE loops, and database access by pressing [Ctrl] + [Break]
- *FASTFK* - permits R:BASE to operate a foreign key index using a condensed index for maintaining that foreign key
- *INDEXONLY* - sets a flag to disable "index only" select retrievals
- *INSERT* - uses either the insert or overwrite mode. Pressing [Ins] toggles you between insert mode and overwrite mode. If this option is not checked, you can use only the overwrite mode.
- *LAYOUT* - saves the layout of the data displayed in the Data Browser (single tables only)
- *MANOPT* - toggles the use of automatic table-order optimization that R:BASE performs when running queries
- MDI - toggles the ability to launch forms, reports, labels, external forms, tables, or views from the Database Explorer or toolbar as a modeless window
- *MESSAGES* - displays system messages
- *NOCALC* - suppresses or processes computed column expressions with the UNLOAD and LOAD commands
- *PAGELOCK* - specifies how R:BASE locks data when updating and deleting rows
- *PASSTHROUGH* - toggles the passing of SELECT statements directly to the foreign data source and are not processed by R:BASE
- *RECYCLE* - toggles the ability for R:BASE to search for a suitable unused block of data within the #2 files, rather than always adding a new block to the end of the file when adding new rows
- *RULES* - enforces data entry rules; R:BASE checks data against all existing rules during data entry and modification. If this option is not checked, R:BASE ignores all rules when rules are not defined for a table, you are archiving data, or you are transferring data into another database.
- *SERVER* - toggles the display messages from a foreign data source
- *SORT* - uses the sort optimizer; R:BASE sorts the minimal amount of data for large tables and recombines the sorted data with the unsorted rows using the minimum amount of disk space.
- *SORTMENU* - displays options in menus in alphabetical ascending order. Menus with column names and values remain unsorted in their original order.
- *UTF8* - supports Unicode characters for string functions in applications and environments which will use higher character sets
- *WALKMENU* - allows users to access menu selections by typing the beginning characters (up to when a match is made) of their names. See also Walkmenu Time Interval
- *WHILEOPT* - improves the optimization and processing of WHILE ...ENDWHILE loops within applications by pre-compiling variables used within the WHILE loop
- *WINAUTH* - support Windows authentication for database connections
- *WINBEEP* - allows R:BASE to access a subset of the sound events in Windows
- *WRAP* - wraps text in fields with NOTE, VARCHAR, and TEXT data types in FILLIN dialogs
- *WRITECHK* - toggles if R:BASE will verify every write to disk
- *ZERO* - treats null values as a zero in a mathematical expression involving INTEGER, NUMERIC, REAL, DOUBLE, CURRENCY, DATE, DATETIME, or TIME data types

- *LOOKUP* - tells R:BASE how many form look-up expressions to store in memory
- *TOLERANCE* - sets the tolerance for comparisons between numbers with REAL and DOUBLE data types
- *NOTE_PAD* - allocates an additional percentage of storage space in NOTE columns to accommodate value increases (additional text), so that rows don't need to move to different disk locations
- *QUALCOLS* - specifies the number of qualkeys to assign to SQL attached tables
- *TIMEOUT* - shuts down an inactive R:BASE workstation and exit to Windows after a set amount of time passes
- *MIRROR* - maintains a duplicate copy of the database

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.



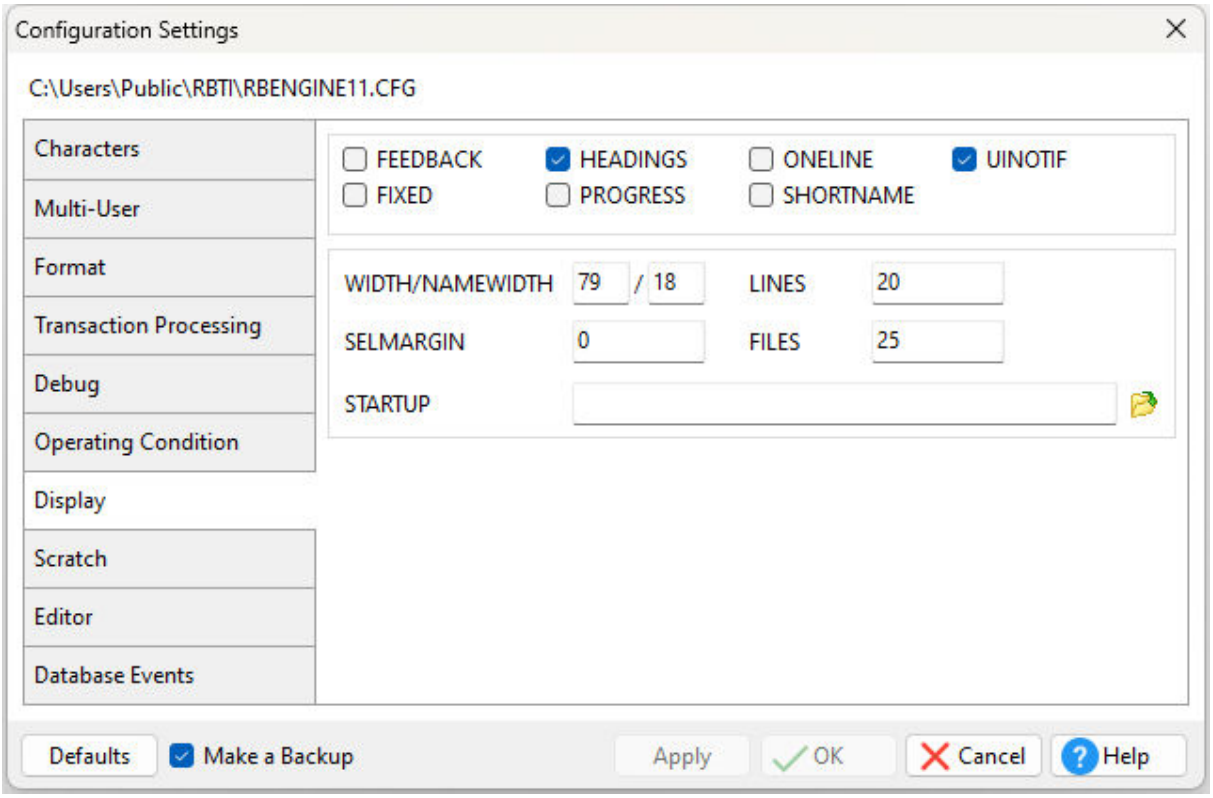
2.22.7 Display

- *FEEDBACK* - displays processing results when either calculating or editing rows. R:BASE displays the number of rows processed and the elapsed time to completion in the status bar. Displays occur while in the Data Browser, when printing reports and labels, and when using certain R:BASE commands.
- *FIXED* - controls whether R:BASE will automatically shrink column widths in SELECT commands
- *HEADINGS* - displays columns with headings when you enter the SELECT or TALLY commands
- *PROGRESS* - displays processing results when building indexes, packing or reloading a database
- *ONELINE* - determines if NOTE and TEXT fields will wrap to the next line in Reports and SELECT commands. Instead they will be truncated at the end of the column.
- *SHORTNAME* - alters the display format of the directory contents, where the file names are listed. With SHORTNAME set to ON, the DIR command lists the file name, extension, size in bytes, and the date and time files were last modified, only listing the contents in the traditional format.

- *WIDTH/NAMEWIDTH* - controls the width of a data line that R:BASE directs to the R> Prompt, printer, screen, or file when using the BACKUP, COMPUTE, CROSSTAB, DISPLAY, SELECT, TYPE, UNLOAD, or WRITE commands
- *SELMARGIN* - set the left margin for displaying the results of a SELECT command
- *LINES* - sets the number of lines per page or screen when you use the CROSSTAB, DISPLAY, DIR, LIST, OUTPUT, SELECT, LIST RULES, SHOW VARIABLES, TALLY, or TYPE commands
- *FILES* - sets the maximum number of files that can be open at a time
- *STARTUP* - specifies a command file to run when R:BASE starts
- *UINOTIF* - automatically displays user interface updates at the Database Explorer. Disabling this option within an application will improve the performance.

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.



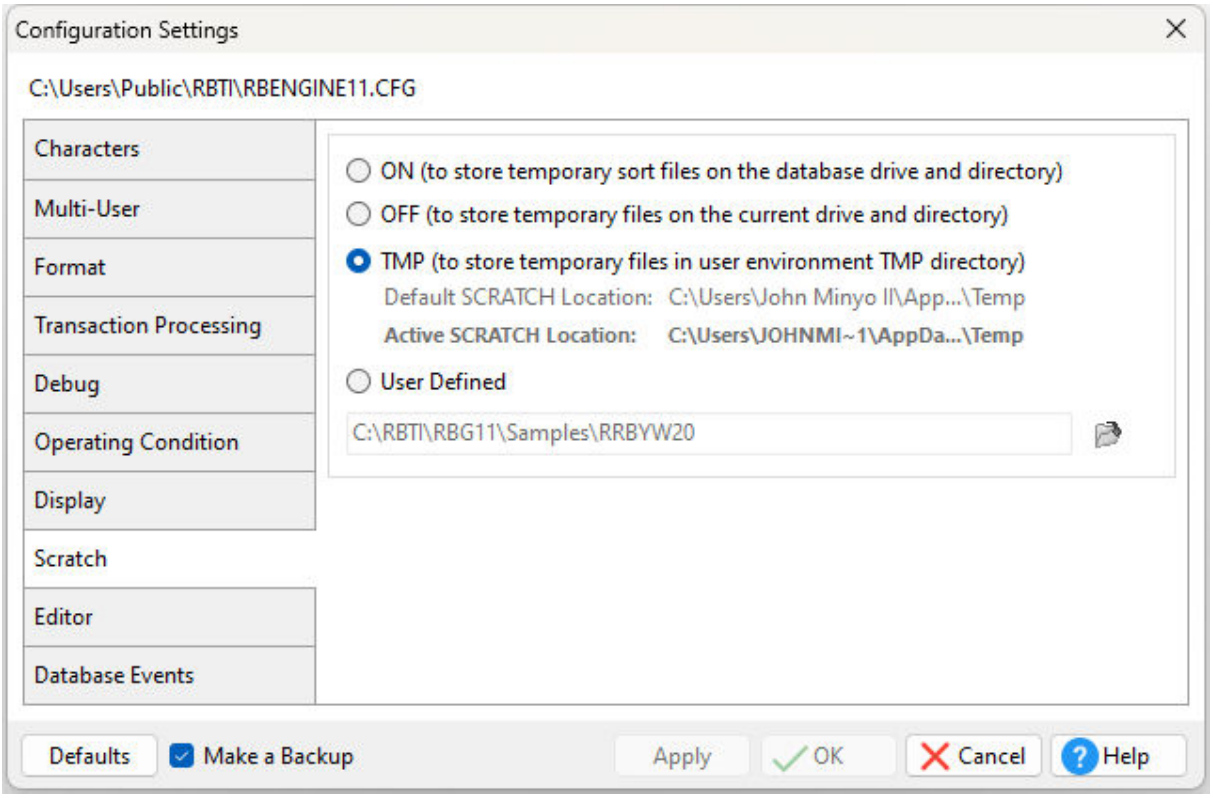
2.22.8 Scratch

The SCRATCH setting determines the drive and directory location for temporary files created when sorting data.

- *ON* - to store temporary sort files on the database drive and directory
- *OFF* - to store temporary files on the current drive and directory
- *TMP* - to store temporary files in user environment TMP directory
- *User Defined* - provides the path to the location where temporary files are stored

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.22.9 Editor

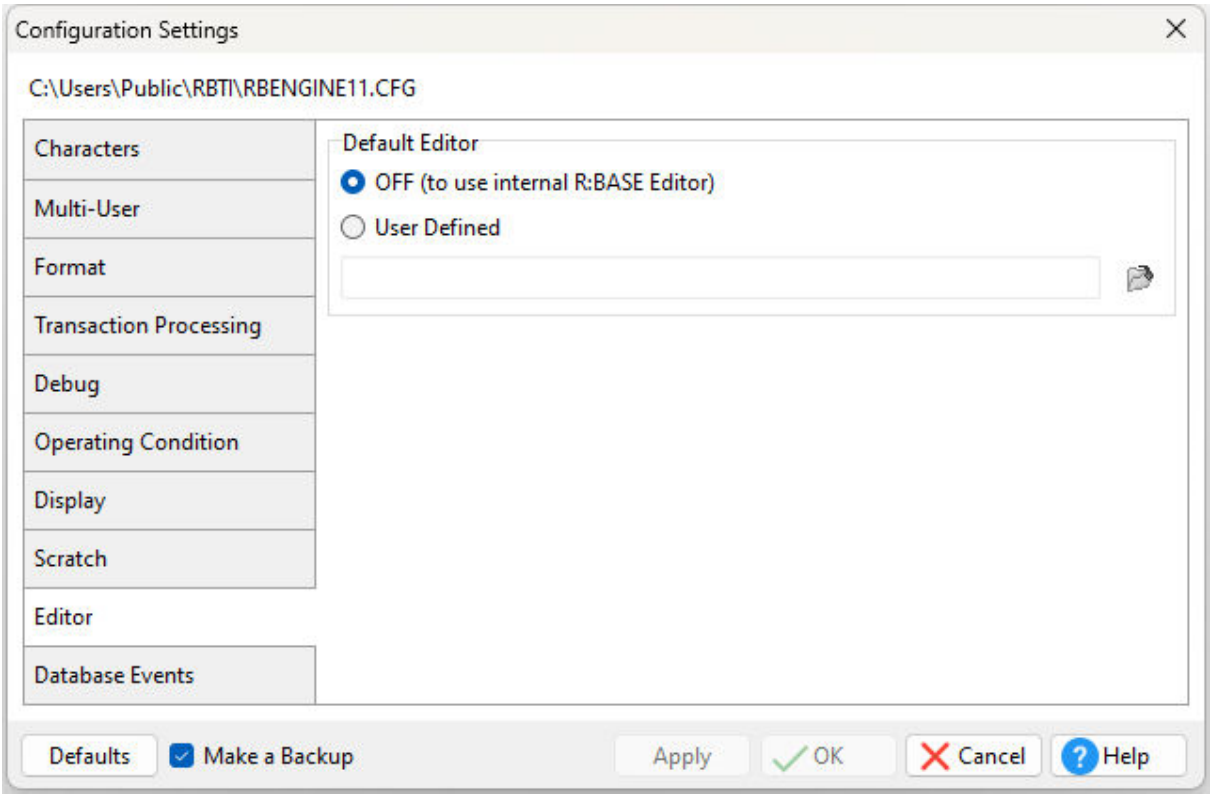
This setting allows you to specify the internal R:BASE Editor or some other text editor as your default text editor for R:BASE command files.

For example, if you wish to alter the default text editor to the external R:BASE Editor program, you would use the following syntax:

```
SET EDITOR C:\RBTI\RBEdit\RBEdit.exe
```

Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.



2.22.10 Database Events

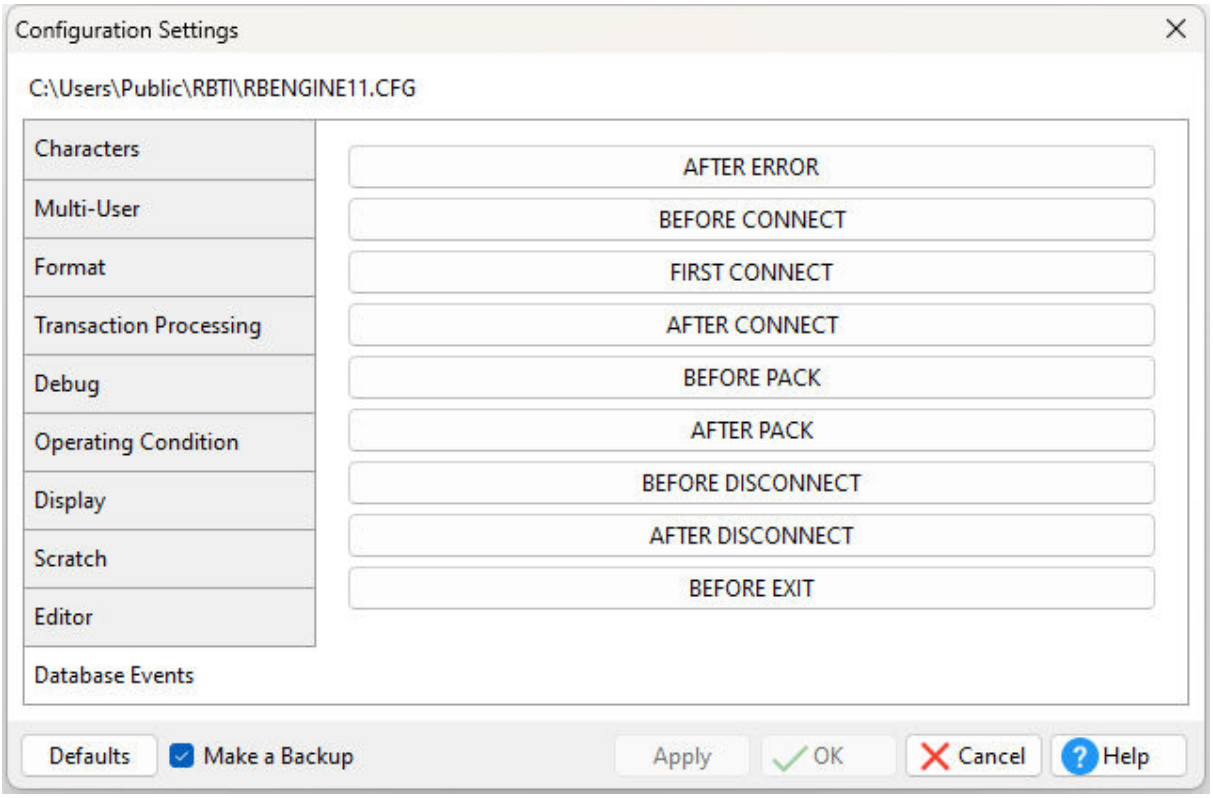
Database events can be used to run a command or command file whenever an event occurs.

- AFTER ERROR - specifies command syntax to run after an error occurs
- BEFORE CONNECT - specifies command syntax to run before a database is connected
- FIRST CONNECT - specifies command syntax to run when the database is first connected
- AFTER CONNECT - specifies command syntax to run after a database is connected
- BEFORE PACK - specifies command syntax to run before a database PACK is performed
- AFTER PACK - specifies command syntax to run after a database PACK is performed
- BEFORE DISCONNECT - specifies command syntax to run before the database is disconnected
- AFTER DISCONNECT - specifies command syntax to run after the database is disconnected
- BEFORE EXIT - specifies command syntax to run before R:BASE exits

A command can be defined to be executed when an event occurs. The RUN command can be defined to run a file, where "Insert File Name" button is available to load a defined file name.

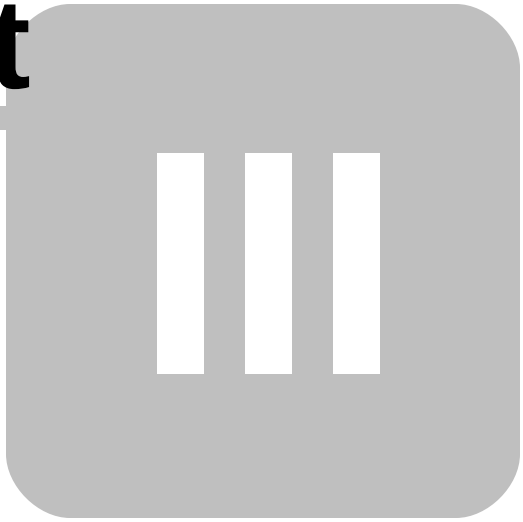
Defaults - will load the default configuration settings for the R:BASE environment
Make a Backup - will create a backup configuration file in the same directory. The backup configuration file will use the .~CFG file extension.

Copyright © 1982-2024 R:BASE Technologies, Inc.



Copyright © 1982-2024 R:BASE Technologies, Inc.

Part



3 Glossary

3.1 A

3.1.1 access rights

Permissions that allows a user to access to a database and tables, and allow the user to assign access rights to other users.

3.1.2 aggregate functions

A function where the values of multiple rows are grouped together to form a single summary value.

Calculates the sum, count, minimum, maximum, or average for a selected column and returns a variable that can be used in a breakpoint in a report. Variables created with aggregate functions are reset after each breakpoint, allowing you to compute subtotals. See also Aggregate Functions

3.1.3 ampersand (&)

(1) A variable used to concatenate characters in a text string with a space.

(2) Variable concatenation character, which indicates that a variable contains an R:BASE command, column, variable, or a combination that needs to be interpreted by R:BASE when the variable is issued in a command line.

3.1.4 ampersand variable

Sets the first variable equal to the exact contents of a second variable; the ampersand tells R:BASE to evaluate the contents of the variable first.

For example, if varname=(2+3), then &varname will be the value 5.

You use dot and ampersand (&) variables differently. A dot variable stores a single value. An ampersand variable holds any portion or all of one command, such as a list of column names or a complete expression. If you store an expression in an ampersand variable, the result of the expression is used in the command.

```
SET VARIABLE vcollist = 'transid, custid, empid'
SELECT &vcollist FROM transmaster
```

The result of the SELECT command is the same as entering:

```
SELECT transid, custid, empid FROM transmaster
```

You can include commands in an ampersand variable since R:BASE interprets the contents of the variable before using it. Thus an ampersand variable could contain all or a portion of a command such as a WHERE clause. However, the variable cannot contain more than one complete command or be used inside parentheses. In addition, you cannot use a semicolon to string together multiple commands within an ampersand variable.

3.1.5 ANSI

(1) Acronym for the American National Standards Institute. References non-Unicode character sets and applications using a graphical user interface on Windows systems.

(2) Reference to non-Unicode characters using the default Windows code page

3.1.6 API

The file extension of the application file generated by the Application Designer, for internal use, in R:BASE versions 4.5++ through 6.5++. The old Application Designer used it to generate the .APP and .APX files.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.1.7 APP

The file extension of the ASCII application file generated by the Application Designer in R:BASE versions 4.5++ through 6.5++. As it was used to review and customize an application command syntax it is still recognized in the latest versions, and can be reviewed under the "Command Files" option within the Database Explorer.

3.1.8 application

A collection of menus, screens, and commands that facilitates completing routine or repetitive tasks.

3.1.9 Application Designer

The R:BASE utility that allows you to build and modify R:BASE applications.

3.1.10 application files

Files with the .RBA which contain the application in version 7.x and higher or files with the .APP, .APX, and .APW/.API extensions, which contain the application in version 6.5++ and lower.

3.1.11 APW

The file extension of the application file generated by the Application Designer, for internal use, in R:BASE for Windows versions up to 6.5++. The old Application Designer used it to generate the .APP and .APX files.

3.1.12 APX

The file extension of the executable binary procedure file generated by the Application Designer in R:BASE versions 4.5++ through 6.5++. The APX file extension is still in use within the R:BASE CodeLock utility.

3.1.13 arguments

Specific information you must provide for a command to work; for example, names of columns or tables, or a list of conditions or values. Arguments are displayed in lowercase letters within the Command Index and represent some specific information you enter in a command.

3.1.14 ASCII

American Standard Code for Information Interchange. A widely used computer character set in which each letter, number, and symbol has a unique numeric value. Most computers can read the ASCII character set. An ASCII file is a file that contains only characters from this character set.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.1.15 ASCII character chart

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Extended Character Chart

Copyright © 1982-2024 R:BASE Technologies, Inc.

DOS	WIN	Dec	Hex	DOS	WIN	Dec	Hex	DOS	WIN	Dec	Hex	DOS	WIN	Dec	Hex
Ç	€	128	80	á		160	A0	└	À	192	C0	α	à	224	E0
ü		129	81	í	¡	161	A1	┴	Á	193	C1	ß	á	225	E1
é	‚	130	82	ó	¢	162	A2	┬	Â	194	C2	Γ	â	226	E2
â	ƒ	131	83	ú	£	163	A3	├	Ã	195	C3	π	ã	227	E3
ä	„	132	84	ñ	¤	164	A4	─	Ä	196	C4	Σ	ä	228	E4
à	…	133	85	Ñ	¥	165	A5	┼	Å	197	C5	σ	å	229	E5
å	†	134	86	ª	¦	166	A6	╞	Æ	198	C6	µ	æ	230	E6
ç	‡	135	87	º	§	167	A7	╟	Ç	199	C7	τ	ç	231	E7
ê	ˆ	136	88	¿	¨	168	A8	╚	È	200	C8	Φ	è	232	E8
ë	‰	137	89	⌐	©	169	A9	╔	É	201	C9	Θ	é	233	E9
è	Š	138	8A	¬	ª	170	AA	╩	Ê	202	CA	Ω	ê	234	EA
ï	‹	139	8B	½	«	171	AB	╦	Ë	203	CB	δ	ë	235	EB
î	Œ	140	8C	¼	¬	172	AC	╠	Ì	204	CC	∞	ì	236	EC
ì		141	8D	¡	-	173	AD	═	Í	205	CD	ø	í	237	ED
Ä	Ž	142	8E	«	®	174	AE	╬	Î	206	CE	ε	î	238	EE
Å		143	8F	»	¯	175	AF	╧	Ï	207	CF	∩	ï	239	EF
É		144	90	░	°	176	B0	╨	Ð	208	D0	≡	ð	240	F0
æ	‘	145	91	▒	±	177	B1	╤	Ñ	209	D1	±	ñ	241	F1
Æ	’	146	92	▓	²	178	B2	╥	Ò	210	D2	≥	ò	242	F2
ô	“	147	93	│	³	179	B3	╙	Ó	211	D3	≤	ó	243	F3
ö	”	148	94	┤	´	180	B4	╘	Ô	212	D4	⌠	ô	244	F4
ò	•	149	95	╡	µ	181	B5	╒	Õ	213	D5	⌡	õ	245	F5
û	–	150	96	╢	¶	182	B6	╓	Ö	214	D6	÷	ö	246	F6
ù	—	151	97	╖	·	183	B7	╫	×	215	D7	≈	÷	247	F7
ÿ	˜	152	98	╕	¸	184	B8	╪	Ø	216	D8	°	ø	248	F8
Ö	™	153	99	╣	¹	185	B9	┘	Ù	217	D9	•	ù	249	F9
Ü	š	154	9A	║	º	186	BA	┌	Ú	218	DA	·	ú	250	FA
¢	›	155	9B	╗	»	187	BB	█	Û	219	DB	√	û	251	FB
£	œ	156	9C	╝	¼	188	BC	▄	Ü	220	DC	ⁿ	ü	252	FC
¥		157	9D	╜	½	189	BD	▌	Ý	221	DD	²	ý	253	FD
Pts	ž	158	9E	╛	¾	190	BE	▐	Þ	222	DE	■	þ	254	FE
ƒ	Ÿ	159	9F	┐	¿	191	BF	▀	ß	223	DF		ÿ	255	FF

3.1.16 ASCII delimited file

An ASCII file in which individual fields in each record (row) are separated with a delimiter character such as a comma (,) or a blank.

3.1.17 ASCII fixed-field file

An ASCII file in which the same field in each record (row) has the same length, regardless of the number of characters in the field. Also called a columnar text file.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.1.18 asterisk (*)

(1) In reports or results from a SELECT command, asterisks indicate a field is too narrow to display a value completely.

(2) Command option requesting that R:BASE use all columns with the command.

(3) In a listing of access rights, an asterisk indicates an access right for which GRANT permission is assigned.

(4) Where a double asterisk (**) is used, the operator becomes exponential, raising a number to a specified power.

3.1.19 AUTOEXEC.BAT

A DOS operating system file that automatically executes when you start the computer.

3.1.20 autonumbered column

A column whose values R:BASE automatically calculates and enters when a row of data is added to the table.

3.1.21 autorefresh

An R:BASE feature that periodically updates data in fields that have been changed by other users.

3.2 B

3.2.1 back up

To make a duplicate copy of your database data and/or structure.

3.2.2 backup file

A file created by backing up all or part of a database. A backup file for a whole database includes both the data and the information needed to recreate the database structure. A backup file for a part of a database includes just the information needed to recreate that part; for example, the data for a single table.

3.2.3 band

A separator that divides report/label sections. The bands appear as gray lines that span across the designers from left to right. The bands can be dragged with the mouse cursor to increase or decrease the size. Specific settings are available for each band based upon its type.

3.2.4 Base64

A group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. The term Base64 originates from a specific MIME content transfer encoding.

Base64 encoding schemes are commonly used when there is a need to encode binary data that needs to be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remains intact without modification during transport.

3.2.5 Before Image File

A Before Image File contains a log of the commands issued in a transaction. The Before Image files are in the same directory as the database and have extensions that begin with "B," such as DATABASE.B01.

3.2.6 Binary-Encoded File

A file encoded in binary format with CodeLock; which uses bits to represent numbers, characters, and symbols.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.2.7 binary string

A composite data type that consists of a length prefix, a data string, and a terminator.

Length Prefix
- Consists of a four-byte integer
- Occurs immediately before the first character of the data string
- Contains the number of bytes in the following data string
- Does not include the terminator

Data String
- Consists of a string of Unicode characters (wide or double-byte characters)
- May contain multiple embedded null characters

Terminator
- Consists of two null characters (0x00)

3.2.8 BLOB

Binary Large Object; binary data that is stored in the database as BIT, BITNOTE, VARBIT, and LONG VARBIT data types. Usually a graphic image

3.2.9 block

A unit of commands or keywords in a procedure file that performs a task. Valid block types are COMMAND, SCREEN, and MENU.

3.2.10 BOM

The byte order mark (BOM) is a Unicode character used to signal the byte order of a text file or stream.

3.2.11 break footer

A report section that corresponds to the end of a breakpoint. At the end of a breakpoint, the break footer is printed, then the breakpoint changes. For instance, you can print a sub-total in a break footer. The abbreviation for the break footer section begins with F0 and increments for every additional break added (e.g. F1, F2, F3, etc.).

3.2.12 break header

A report section that corresponds to the beginning of a breakpoint. At the beginning of a breakpoint, the break header is printed, then the corresponding detail information is printed. For instance, you can print a sub-heading in a break header. The abbreviation for the break header section begins with H0 and increments for every additional break added (e.g. H1, H2, H3, etc.).

3.2.13 break variable

A variable that causes a break in a report when the value of the variable changes.

3.2.14 breakpoint

(1) Any command in a command file where the developer wants to stop processing when tracing the command with the interactive Trace Debugger.

(2) In reports, a breakpoint, or break, indicates that values are to be grouped and sorted by a specified column or variable.

3.2.15 browse

To view data in the "Data Browser" window.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.2.16 buffer

Temporary space allocated in memory to perform computations.

3.2.17 byte

Representations of words and numbers; one text character is equivalent to eight bits, or one byte. The storage capacity of computers is represented in bytes, such as 640K (kilo) bytes. A kilobyte is 1,024 bytes.

3.3 C

3.3.1 Case Folding Table

A section in the configuration file that contains a list of corresponding ASCII character codes for uppercase and lowercase letters. The case folding table allows R:BASE to test characters for equality when CASE is set off in operations such as IF and WHILE commands and WHERE clauses.

3.3.2 cascade

Maintains primary/foreign key relationships automatically. For example, if you either update or delete a primary key value from this table, the corresponding foreign key values are updated or deleted automatically.

3.3.3 CFG

The file extension for the R:BASE configuration file.

3.3.4 character

A single printable letter, numeral, or symbol used to represent data. Includes hidden characters such as space, tab, and carriage return. In R:BASE, the data type for characters is TEXT.

3.3.5 Character Folding Table

A section in the configuration file that lists ASCII character codes that relate uppercase characters to lowercase characters. The character folding table allows R:BASE to relate uppercase characters and lowercase characters in string manipulation functions.

3.3.6 character modifier

Replaces, adds to, or modifies individual characters or numbers within a field.

3.3.7 check box

A box on an R:BASE form, which the user can check or uncheck, corresponding to a yes/no or true/false selection.

3.3.8 clickable image

An image placed on a form that acts the same as a push button. When a user clicks on the image a predefined action or entry/exit procedure is run.

3.3.9 CodeLock

The R:BASE utility used to convert ASCII screen, menu, command, and procedure files to binary command or procedure files. See CodeLock.

3.3.10 Collating Table

A section in the configuration file that contains a list of corresponding ASCII character codes for use in sorting and equality testing. Each line in the table begins with the word COLLATE and lists two ASCII character codes. R:BASE considers the first ASCII character code to be equivalent to the second in

Copyright © 1982-2024 R:BASE Technologies, Inc.

sorting sequence. The collating table includes characters whose codes would make the sorting sequence inaccurate (for example A and a).

3.3.11 column

A specific piece of information defined for a database table. In R:BASE and other products, a column is also called a field or data field.

3.3.12 column object

A field you place on a form, report, or label that corresponds to a value in a column. In forms, you can enter, modify, or display a column value; in reports or labels, the column object displays a column value.

3.3.13 Columnar Text File

See ASCII Fixed-Field File.

3.3.14 combo box

A drop-down list on an R:BASE form. Non-editable combo boxes allow the user to select a value; editable combo boxes allow the user to enter or select a value.

3.3.15 command

An instruction to the computer, entered from the keyboard or a file. A command can be a word, abbreviation, or character that directs the system to perform a predefined operation.

3.3.16 command block

A group of commands you store for reuse in a procedure file. Identify the command block with the $COMMAND label and the name of the command block.

3.3.17 command file

A logical set of commands forming a single program and stored in an ASCII file. It cannot contain the $COMMAND identifier. R:BASE executes the commands contained in the command file when the file is run using the INPUT or RUN commands.

3.3.18 command name

The full name or abbreviation of an R:BASE command, such as SELECT, SET VAR, and RBEDIT. Command names are shown in syntax diagrams in uppercase letters.

3.3.19 COMMAND.INI

The COMMAND.INI file includes the most commonly used pre-defined syntax for R:BASE commands and Functions.

While working with command files in the R:BASE Editor, the COMMAND.INI file is the source for the "List of Command at Cursor" and "Build Command at Cursor" syntax utilities located under "Structure" on the Menu Bar.

Simply type the first few characters of the R:BASE Syntax, and then press the [F5] key to have the command syntax typed automatically for you. Or, press [Ctrl]+[F5] to build the syntax before placing it into the command file.

You can also customize the COMMAND.INI file, using R:BASE Editor, according to your favorite style of most commonly used commands.

The file is installed in your R:BASE program directory.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.3.20 comments

(1) Text in command or application files to provide internal program documentation. The comment text is ignored when R:BASE processes the program

(2) Descriptions for tables, columns, forms, reports, etc.

3.3.21 common column

A column in two or more tables that has the same name, data type, and if TEXT or NUMERIC, the same size. See also Linking column.

3.3.22 comparison operator

Compares one item to another in an expression or WHERE Clause. Examples of comparison operators are: >, <, =, <>, BETWEEN, LIKE, and CONTAINS.

3.3.23 computed column

A column that uses an expression or function to calculate its value from the values of other columns in a table or from constants you specify.

3.3.24 concatenate

To link text values together in a series.

3.3.25 concurrency control

An R:BASE feature that controls multiple access to tables when users are using forms or the EDIT commands in multi-user mode.

3.3.26 condition

A condition allows you to tell R:BASE to retrieve only the rows of data that meet specific criteria. Conditions can include a combination of one or more expressions and/or operations that would evaluate to either true or false. Conditions are used in WHERE clauses and in the IF, WHILE, and SWITCH commands.

3.3.27 condition list

Lists a set of conditions that combine to form a statement that is either true or false. Conditions are combined with the connecting operators AND, OR, AND NOT, and OR NOT.

3.3.28 configuration file

The file used to set the R:BASE operating environment.

See also:

RBENGINE11.CFG

3.3.29 connecting operators

Words that connect conditions in a WHERE clause. The connecting operators are AND, AND NOT, OR, and OR NOT.

3.3.30 constraint

A data integrity restriction that limits or validates the data in a column to ensures that incoming data is compatible with existing data. Constraints are commonly used to refer to primary and foreign keys. R:BASE provides the following constraints: primary key, foreign key, unique key, and not null. See Constraints.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.3.31 continuation character

A specific character that allows you to type single entries on multiple lines. The default continuation character is the plus sign (+).

A plus character followed by the greater than sign indicates that R:BASE expects a continuation of the command on the previous line. (+>)

3.3.32 control menu

Also called the system menu, the control menu appears when you click on the horizontal bar in the upper left comer of a window. Used to size or close a window.

3.3.33 correlation name

A nickname or alias for a table or view name; allows you to refer to a table or view in a command without having to retype the name each time it is used.

3.3.34 cursor

(1) On the screen, a movable marker or line that designates the next point of character entry or change, or is used to select items or toolbar buttons.

(2) In a command file, a pointer to a row in a table or view. See DECLARE CURSOR.

3.3.35 custom action

An action assigned to a menu option, which you create by writing a command block. Used in the Form Designer and Application Designer.

3.3.36 custom EEP

A series of commands embedded with a form, report, or label. The only difference between an EEP and a Custom EEP is that the Custom EEP is stored in the form, report, or label, whereas a regular EEP must be stored in a command file with the .EEP file extension. See EEP.

3.4 D

3.4.1 DAT

The file extension for an ASCII-delimited file exported from R:BASE.

3.4.2 Data Browser

The R:BASE utility for viewing data in tables in a tabular format. If you want to edit data in this format, switch to the edit mode (Data Editor).

3.4.3 Data Editor

The Data Browser in edit mode; you can then edit data in a tabular format. To view data only, switch to browse mode.

3.4.4 data entry mask

A control to limit which characters can be accepted from the keyboard when entering or editing data in a form.

3.4.5 data entry rule

A user-defined regulation that controls the entry and modification of data in a column or a group of columns.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.4.6 data integrity

The concept of ensuring that data from one table relates to data in other tables, so that data that is added to related tables always matches.

3.4.7 data set

A data set (or dataset) is a collection of data.

3.4.8 data source

A data source is an R:BASE database or a database created with a different program.

3.4.9 data type

A specification of the nature of the data held in a column or variable. For a description of each data type, see Data Types.

3.4.10 database

A collection of logical groups of data, stored in one or more files, used to perform a task.

3.4.11 Data Designer

The R:BASE utility that allows you to create and modify databases, columns, tables, constraints, rules, and indexes.

3.4.12 database files

Files which contain the database structure, data, indexes, and large binary data. For R:BASE 11, the extensions are .RX1, .RX2, .RX3, and .RX4.

3.4.13 database lock

In multi-user mode, a lock on the entire database preventing any changes to structure or data.

3.4.14 DBMS

Database Management System. A means for storing and retrieving information in a systematic order. Libraries, dictionaries, and file cabinets are all examples of database management systems.

3.4.15 debugger

See Interactive Debugger.

3.4.16 default settings

Options that R:BASE sets automatically. You can change default settings. For example, you might want to change the default font setting for a form.

3.4.17 delimiter

A character that marks the end of a unit of data or separates items in a list. The character also separates data fields in an ASCII delimited file. The R:BASE default delimiter is a comma (,).

3.4.18 descending order

Sorting from highest to lowest (such as 50 to 1, Z to A). Sorting follows the reverse order of ASCII character codes.

3.4.19 detail section

The report section that repeats for each row of data drawn from the database. The abbreviation for the Detail section in a report is the letter D.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.4.20 determinate

A value that allows you to obtain another value.

3.4.21 device

Term used to describe the current entry source or output destination; for example, the keyboard or a printer.

3.4.22 dialog box

A text-based or object-based box for user input or responses.

3.4.23 dimmed

When a menu option is unavailable, R:BASE displays it in a different color or intensity.

3.4.24 directory

A subdivision of the disk space.

3.4.25 display mask

Modifiers that control display and printing of characters in a field.

See also: Character Modifiers and String Modifiers

3.4.26 DOS

Disk Operating System. Short form of PC-DOS or MS-DOS.

3.4.27 dot variables

A dot (or period) or a colon in front of a variable name indicating that R:BASE will use the value of the variable.

3.4.28 double asterick (**)

Exponential operator. Raises a number to a specified power.

3.4.29 drive

The computer component used to read and write data on magnetic storage devices such as disks.

3.4.30 drive specification or designation

A letter assigned to a logical or physical division of a disk drive system. For example, drive a:.

3.4.31 driving table or view

The table or view that is the primary source of data for a report, form, or label. Each can receive data from several tables; specify the driving table or view to be the table from which the most data is gathered.

3.4.32 DUP

The value #DUP can be placed in the "Default Value" field for DB Edit controls to duplicate the value when entering multiple rows of data through forms.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.5 E

3.5.1 echo

An R:BASE setting that displays commands in a command file to the output device as the commands are executed. See SET ECHO.

3.5.2 EEP

(1) An EEP can represent a command file that uses the .EEP file extension, or can represent a series of commands embedded with a form, report, or label, which is then referred to as a "Custom EEP". The only difference between an EEP and a Custom EEP is that the Custom EEP is stored in the form, report, or label, whereas a regular EEP must be stored in a command file with the .EEP file extension.

(2) The file extension used for entry/exit procedures.

3.5.3 endkey

The final keystroke that the user presses in response to a dialog box (e.g. [Enter] or [Esc]).

3.5.4 entry/exit procedure (EEP)

Entry/Exit Procedure. Entry/Exit Procedures (EEP) are series of commands that can be run within forms, reports, and labels.

An EEP can represent a command file that uses the .EEP file extension, or can represent a series of commands embedded with a form, report, or label, which is then referred to as a "Custom EEP". The only difference between an EEP and a Custom EEP is that the Custom EEP is stored in the form, report, or label, whereas a regular EEP must be stored in a command file with the .EEP file extension.

An EEP can also be a series of commands that can be executed within a command block of a binary codelocked application file.

EEPs provide numerous objectives for versatility in controlling data processing and screen displays.

See Also:

Entry/Exit Procedures (EEPs)

3.5.5 environment settings

The settings for the conditions, special characters, multi-user features, and data entry and display formats that determine the R:BASE operating environment.

3.5.6 error messages

Text displayed by the computer when an entry is incorrect or some other condition prevents the system from carrying out a command.

3.5.7 error variable

A variable that holds the status of the current operation: 0=success, a positive number=failure.

See also: SQLCODE

3.5.8 exclusive lock

A table lock obtained through the SET LOCK command. The table remains locked until the SET LOCK OFF command is issued.

3.5.9 Expansion Character Table

A section in the configuration file that lists ASCII character codes for up to seven characters to be equivalent to two other characters.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.5.10 explicit data typing

To define a data type using the SET VARIABLE command.

3.5.11 exponential operator (**)

Arithmetic operator. Raises a number to a specified power.

3.5.12 export

To send data from R:BASE to a file for use with another product.

3.5.13 exporting data

Transferring data from R:BASE to an external file.

3.5.14 expression

A text or arithmetic formula that produces a single value as the result. An expression can include columns from a table, constant values, functions, or system variables such as *#date*, *#time*, and *#pi*.

3.5.15 External Form File

An R:BASE form that is stored outside the database. External Form Files use the .rff file extension. External Form Files can be created and modified using the External Form Designer.

3.6 F

3.6.1 field

Areas on the screen where data is entered or displayed. A field can pertain to an object on a form, report, or label that corresponds to a column in a table or view. Fields display database information from the corresponding column. In a form, you can also enter and edit database information in fields.

3.6.2 file

Data that is stored on a disk and given a unique file name. An R:BASE 11 database is stored in four operating system files: DBNAME.RX1, DBNAME.RX2, DBNAME.RX3, and DBNAME.RX4, where DBNAME is the name of the database.

3.6.3 file extension

The three letters that follow the period after a file name according to the naming conventions of the operating system.

3.6.4 file handle

A number used to reference an internal table that the operating system uses to keep track of currently open files.

3.6.5 filespec

File specification. A filename including drive and directory specification. A file specification contains these elements:

- A letter that identifies the drive
- One or more names of directories separated by backslashes
- A filename

3.6.6 foreign data source

An R:BASE database other than the one currently open, or a database created with a different program. See also Foreign Data Sources and ODBC

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.6.7 foreign index

With the FASTFK setting on, creates a foreign key that has an index using row pointers for data retrieval on selected columns.

3.6.8 foreign key

A column or set of columns that match values in a particular primary key or unique key constraint defined in a different table. Used to ensure that only valid data is entered into a column.

3.6.9 form

A screen display for editing, entering, and displaying database information.

3.6.10 Form Designer

The R:BASE utility that allows you to create and modify R:BASE forms.

3.6.11 function

A predefined complex expression to which you pass one or more values and which returns a single value. See Functions.

3.6.12 function keys

Keys used for display and editing functions. The function keys are labeled F1, F2, F3, and so on.

3.7 G

3.7.1 global variable

A temporary storage area used to hold a single value. Global variables can be used in expressions, command files, forms, and reports. They can also be used in conditional command processing and to transfer data between tables. A global variable is not associated with any database.

3.7.2 GROUP BY clause

Groups data for display and performs computations using the SELECT functions. For more information, see GROUP BY.

3.8 H

3.8.1 hashing

The process of converting text values longer than a specified length to a numeric representation when creating indexes.

3.8.2 HAVING clause

Limits the rows affected by the GROUP BY clause. For more information, see HAVING.

3.9 I

3.9.1 implicit data typing

A data type that is implied by the value assigned to a variable.

3.9.2 import

To bring data into R:BASE from a file prepared with another product.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.9.3 importing data

Adding data to the database from an external file.

3.9.4 index

A cross-reference for values contained in a table; an index allows R:BASE to find values in faster.

3.9.5 indexed column

A column that contains locations for the data stored in that column. The index speeds searches for data. Columns that are indexed should contain a unique or at least a nearly unique value for every row in the table.

3.9.6 indicator variable

A variable that is used to indicate if a null value is retrieved that used with an INTO clause in a SELECT command. This variable stores the status of the variable: non-null (0) or null (-1).

3.9.7 input device

A device attached to the computer, such as the keyboard or a disk drive, from which data and commands can be read.

3.9.8 input file

(1) A file that holds data and/or commands that can be read and acted on by R:BASE.

(2) A file to be imported by the Import/Export utility.

3.9.9 instruction

A command that tells the computer which operation to perform.

3.9.10 interactive

Two-way communication between a user and computer software. For example, the user issues a command and R:BASE responds.

3.9.11 Interactive Debugger

The R:BASE utility that uses the TRACE command to debug files. The Interactive Debugger allows you to step through code one line at a time to verify that the proper actions result.

3.9.12 interval

Time period, in tenths of seconds, during which a waiting command checks for availability of resources in multi-user mode.

3.10 J

3.10.1 join

A SQL clause that combines records from two tables.

See Table Joins

3.11 K

3.11.1 key

One or more columns that uniquely identify a row. A key can be a primary key, foreign key, unique key, or index.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.11.2 key maps

User-defined keys for storing any series of key strokes you want to use repeatedly in R:BASE.

3.11.3 keyword

A required or optional element of the syntax of the R:BASE command language. Some keywords link the command and the arguments. Others refine the operation of the command. Keywords are shown in syntax diagrams in uppercase letters.

R:BASE reserves some keywords for its use. Do not use those keywords for column, table, view, variable, form, or report names. For more information, refer to Reserved Words.

3.12 L

3.12.1 label

(1) Lines of printed data, such as address information used for mailing labels.

(2) A specification for a line in a command file to which the GOTO command can pass control.

3.12.2 Label Designer

The R:BASE utility that allows you to create and modify R:BASE labels.

3.12.3 LAN

See Local Area Network.

3.12.4 lasso

A freehand tool, for the mouse, that appears as a dotted line while you click and hold the left button while dragging the cursor across the screen. It works by starting in one corner of the work space, holding down the left mouse button, and dragging the cursor to the opposite corner. The lasso only needs to be touching the object in order to select it for editing.

3.12.5 linking

Connecting related information that is stored in different tables.

3.12.6 linking column

A column in one table that contains the same values as a column in one or more other tables. Linking columns can be two columns with the same name, data type, and size (if TEXT or NUMERIC); or, two columns with different names but compatible data types.

3.12.7 LOB

Large Object; large text data stored in the database as VARCHAR or LONG VARCHAR data types. Can be up to 256 MB.

3.12.8 local area network

A hardware and software system that allows workstations to share computer resources.

3.12.9 local lock

In multi-user mode, a local table lock results from a SET LOCK command issued at that workstation.

3.12.10 lock

Prevents one user's command from changing the structure or data of a table or database while another user's command is using the same resource.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.12.11 look-up expression

A method for taking a value from a table for use in a form. There are two types of look-up expressions: standard and master. Standard look-up expressions find values in other tables; master look-up expressions find values in the current table.

3.13 M

3.13.1 many-to-many relationship

Specifies that table 1 can contain many rows for any given row in table 2 and vice versa.

3.13.2 MAPI

MAPI (Messaging Application Program Interface) is a Microsoft Windows program interface that enables you to send e-mail from within a Windows application and attach documents.

3.13.3 mask

(1) Data entry mask; a control to limit which characters can be accepted from the keyboard when entering or editing data in a form.

(2) Display mask; character and string modifiers that replace, add to, modify, center, or justify characters in fields when data is printed or displayed.

See also: Character Modifiers and String Modifiers.

3.13.4 MDI

Specifies a modeless window--a modeless window allows you to access other windows without closing the current window first. (MDI = Multiple Document Interface)

See SET MDI

3.13.5 MDX

The file extension for a multiple index file for dBASE IV, which is not supported by R:BASE.

3.13.6 memory

In a computer, temporary storage area that holds a program and data when a program is run. Also known as RAM, or Random Access Memory.

3.13.7 menu

List of options from which to choose.

3.13.8 menu bar

A menu bar is a horizontal strip that contains lists of available menus for a certain program. In Windows programs, the menu bar resides at the top of each open window.

The R:BASE Menu Bar contains available menus from which you can choose commands, productivity tools, utilities, settings, module-specific options, and help.

3.13.9 menu block

An R:BASE menu definition that you store in a procedure file to display a single menu. Identify it with the label $MENU and run it with the CHOOSE command.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.13.10 menu file

Contains the information and structure used by the CHOOSE command to display a menu on the screen. CodeLock can add a menu file as a menu block in a procedure file.

3.13.11 menu tree

A sketch that shows the order and hierarchy of menus in an application.

3.13.12 menuname

A user-defined menu can be stored either in an ASCII disk file or as a menu block in an R:BASE procedure file created by CodeLock or the Application Designer. If *procfile* is used, *menuname* refers to a *procfile* menu block. If *procfile* is not used, *menuname* specifies an ASCII disk file containing the menu to be displayed.

3.13.13 messages

Information displayed on screen or written to a file that is relevant to the task being performed; for example, error messages or status messages.

3.13.14 MS-DOS

Operating system developed by Microsoft for IBM personal computers and similar machines.

3.13.15 multi-user mode

Enables more than one workstation on a network to use R:BASE at the same time and to share the same database.

3.14 N

3.14.1 NDX

The file extension for a dBASE index file.

3.14.2 nesting

Locating an item within a similar item. In R:BASE, you can nest applications, command files, control structures, functions, and procedure files.

3.14.3 null

A representation consisting of one to four characters that identifies the absence of data or when a value does not apply. The R:BASE default is *-0* -.

3.14.4 numeric data types

Data type formats that enable you to manipulate numeric values using arithmetic operators. The following are numeric data types: CURRENCY, DOUBLE, INTEGER, NUMERIC, and REAL.

3.15 O

3.15.1 object

An item (text, column, or variable) placed on a form, report, or label.

3.15.2 ODBC

Open Database Connectivity (ODBC) is Microsoft's initiative for a standard SQL interface to database products.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.15.3 one-to-many relationship

Specifies that *table 1* can contain only one row for any given row in *table 2* but *table 2* can contain many rows for any given row in *table 1.*

3.15.4 one-to-one relationship

Specifies that *table 1* can contain only one row for any given row in *table 2* and vice versa.

3.15.5 online HELP

Information displayed to assist the user. You can access Online Help in R:BASE by pressing [F1]. For context-sensitive help, press [Shift]+[F1].

3.15.6 operand

A column, variable, string, or value acted on by an operator and combined with another operand to form an expression. In the expression *(x + y)*, *x* and *y* are operands and + is the operator.

3.15.7 operating system

A collection of computer programs that control the overall operation of a computer and perform such tasks as assigning memory to programs and data and controlling the overall entry and output of the system.

See also:

Windows
MS-DOS
OS/2
PC-DOS

3.15.8 operator

A symbol (such as +, -, *, /, **, &) representing an operation to be performed on one or more columns, values, variables, or strings. R:BASE provides the following types of operators: arithmetic, comparison, connecting, expression, and location.

For the symbols: (+, -, *, /, **, &) the mathematical calculations, in the same order, are: addition, subtraction, multiplication, division, exponentiation, and concatenation.

3.15.9 operating system commands

R:BASE commands that manage files, directories, and drives: CHDIR, CHDRV, CHKDSK, COPY, DIR, ERASE, MKDIR, RENAME, RMDIR, TYPE.

3.15.10 optimizer

An internal algorithm that determines the best order for joining tables.

3.15.11 optional parts of the syntax

The optional portion of a command appears below the main line.

3.15.12 ORDER BY clause

Sorts rows of data. For more information, see the ORDER BY command.

3.15.13 orphaned data

Data in one table that does not relate to any data in any other table.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.15.14 OS/2

Multi-tasking operating system originally developed by Microsoft for use on IBM PC-AT and Personal System/2 computers and 100% compatibles; now owned by IBM.

3.15.15 output device

A computer screen, printer, or disk drive, which is attached to the computer and to which data can be sent.

3.15.16 output file

A file to which data or commands are written from R:BASE or the Import/Export utility.

3.15.17 owner

The owner of a database. Refers to the user identifier who has been designated as the owner of the database.

3.15.18 owner identifier

The identifier (an assigned name by which a user can access a database) for the user who has been designated the owner of the database.

3.16 P

3.16.1 pack

To compress a database to free disk space.

3.16.2 padding

Additional percentage of storage space in NOTE columns to accommodate additional text.

3.16.3 page footer

A report section that appears at the bottom of every page of the report. The abbreviation for the page footer section is PF.

3.16.4 page header

A report section that appears at the top of every page of the report. The abbreviation for the report header section is PH.

3.16.5 parameter

A value that is passed to a command specifying actions or files that the command will execute.

3.16.6 parenthesis ( )

Expression identifiers. Enclose functions in parentheses when you use a full expression or when you embed one expression inside another. Use parentheses in WHERE clauses around conditions you want R:BASE to evaluate before other conditions; otherwise, R:BASE evaluates conditions from left to right.

3.16.7 parse

The way R:BASE examines and breaks a command line into its component parts before executing it.

3.16.8 partial text index

An index created for a TEXT column by specifying a number of characters to preserve for comparison and storing the remainder of the text as a hashed value.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.16.9 password

A character sequence assigned to a specific user identifier. A password prevents other users from accessing the database if they know the user identifier.

3.16.10 path

The list of drives and directories to search when looking for executable files.

3.16.11 PC-DOS

Operating system for the IBM PC, PC-XT, PC-AT, and 100% compatibles.

3.16.12 percent variables

System variables that hold the parameter values passed into the RUN command.

For example, the following command would run the file 'command.rmd' and pass in the values 'value1' and 'value2'.

```
RUN command.rmd USING value1 value2
```

In the above example, while command.rmd was executing, value1 could be accessed as %1 and value2 could be accessed as %2.

The percent variables may also be created when using an EEP, as this is technically the equivalent to running the EEP. If you wish to do this then you will need to use a syntax such as the one show here.

3.16.13 picture format

An R:BASE feature that gives you the ability to create display masks for each field on a report. No matter how data is stored in a table, you can print it according to the display mask set in the picture format.

3.16.14 plugin

A file containing data used to alter, enhance, or extend the operation of R:BASE. R:BASE Plugins are made available through different means. Some Plugins are included with the purchase of R:BASE. These files have the .rbl or .rbm file extension. Other Plugins are included with R:BASE Technologies, Inc. add on products such as R:Charts, RBZip, and R:Spell Checker.

Many other software products use plugins. This technology is based on API, which is declared by the program creator. A sample of products which use plugins include: WinAMP, Adobe Photoshop, ACDSee, etc. Support for plugins mean that the computer programmers can enhance and improve product possibilities using programing languages such as C/C++, Pascal, Delphi, and Basic.

3.16.15 plus sign prompt (+>)

Indicates that R:BASE expects a continuation of the current command.

3.16.16 pop-up menu

A menu for a field object to display information for a field. R:BASE supports four types of pop-up menus: single column, multi-column, expression, and expression with a column.

3.16.17 precision

The total number of digits in a NUMERIC data type.

3.16.18 preview

Click this button to look at the layout of your form or label. In the Report Designer, the preview button will show only the first 10 rows.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.16.19 primary key

A column or set of columns that uniquely identify a row; in other words, each value in a primary key column is unique. A primary-key constraint prevents duplicate (non-unique) and null values from being entered into a column. Even if you do not specifically define a constraint, all tables (in a well-designed database) should have a primary key. You can define one primary key per table.

3.16.20 procedure file

A binary file generated by CodeLock containing a series of command blocks forming a complete executable program. The file is normally assigned with an .APX extension.

3.16.21 program

A series of R:BASE commands, grouped in logical sections and stored in either binary or ASCII format, each of which performs a specific task.

3.16.22 pull-down menu

A menu that displays options across a menu bar at the top of the screen. Each option on the menu bar can display a pull-down submenu, and each submenu can display a pop-up submenu.

3.16.23 push button

A button on an R:BASE form that when pressed, corresponds to a predefined action or an entry/exit procedure.

3.17 Q

3.17.1 QBE

See Query By Example.

3.17.2 qualkey

A column definition assigned to SQL attached tables, for columns that uniquely identifies a row.

See also:

QUALCOLS
ODBC
SATTACH

3.17.3 query

A question constructed to get information from database tables and columns. R:BASE provides the Query By Example utility, which allows you to form queries to your database.

3.17.4 Query By Example

An R:BASE utility that allows you to form queries to your database.

3.18 R

3.18.1 RBA

The file extension for R:BASE application files created using R:BASE versions 7.0 and higher. See Application Designer.

3.18.2 R:BASE

The R:BASE Technologies Inc database management system incorporating SQL and R:BASE commands.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.18.3 R>

The R:BASE command prompt in the "R> Prompt" window. See also: R> Prompt

3.18.4 R> Prompt

The R:BASE utility that allows you to enter R:BASE, SQL, and DOS commands.

3.18.5 radio button

A type of button on an R:BASE form. Radio buttons provide a group of options for the user. Only one radio button can be selected at a time.

3.18.6 RB1

The file extension for a database file created using R:BASE versions 4.5 up to R:BASE X.5. This file contains the database structure.

3.18.7 RB2

The file extension for a database file created using R:BASE versions 4.5 up to R:BASE X.5. This file contains the data for the database.

3.18.8 RB3

The file extension for a database file created using R:BASE versions 4.5 up to R:BASE X.5. This file contains the database indexes.

3.18.9 RB4

The file extension for a database file created using R:BASE versions 4.5 up to R:BASE X.5. This file contains binary and text large objects.

3.18.10 RBASE.DAT

An optional R:BASE command file that executes from the current directory after the configuration file when R:BASE starts.

3.18.11 RBDefine

The Data Designer.

3.18.12 RBEdit

The R:BASE Editor; allows you to create and modify text files.

3.18.13 RBENGINE11.CFG

The R:BASE 11 configuration file, read when you start R:BASE, is used to set the default R:BASE operating environment including multi-user operation, environment settings, startup file, key map definitions, and character tables.

3.18.14 RBF

The file extension for databases created using R:BASE 3.0, R:BASE for DOS, and R:BASE System V.

3.18.15 RBS

File extension for databases created with R:BASE 5000 and R:BASE 4000.

3.18.16 RBX

The file extension for databases created with R:BASE 4.0 up to 7.6. "X" is a number from one to four.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.18.17 record

One row in a database table; an individual collection of information.

3.18.18 region

An area in a form that contains all of the located fields related to a single table. You can use a region to display more than one row of data.

3.18.19 relational commands

Commands used to create a new table from existing tables. The relational commands include: INTERSECT, JOIN, PROJECT, SUBTRACT, and UNION.

3.18.20 relational database

A collection of information in which data is stored in a set of tables; linking columns among the tables allow for multiple-table operations so it can be expanded, updated, and retrieved rapidly

3.18.21 relational tools

Operations used to create new tables or append data to existing tables. Relational tools are INTERSECT, JOIN, PROJECT, SUBTRACT, and UNION.

3.18.22 remote lock

In multi-user mode, a table lock resulting from a SET LOCK command issued from another workstation on the network.

3.18.23 repeatable parts of the syntax

An arrow in a command syntax indicates what portion of the command can be repeated. Each part of the command that is repeated must be separated with a comma, or the current delimiter character.

3.18.24 report

A format for retrieving information to be sent to a file, printer or screen.

3.18.25 Report Designer

The R:BASE utility that allows you to create and modify R:BASE reports. See Report Designer.

3.18.26 report footer

This section of a report displays information once at the end of the report, such as a report's grand total. The abbreviation for the report footer section is RF.

3.18.27 report header

This section of a report displays information once at the beginning of the report, such as the report's title. The abbreviation for the report header section is RH.

3.18.28 report section

Parts of a report, such as the report header and footer, the page header and footer, the detail section, and break headers and footers.

3.18.29 required parts of the syntax

Required parts of the command are on the main line of the syntax.

In this syntax example, the brackets are also on the main line, therefore the syntax requires you to make a choice from the keywords and arguments.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.18.30 reserved words

Commands, keywords, and other names used exclusively by R:BASE. Do not use these words or any shortened form of them as names for tables, columns, or anywhere else in R:BASE. See Reserved Word List.

3.18.31 restore

To use a backup copy to replace lost or damaged data, the database structure, or both.

3.18.32 RIM

Relational Information Management. RIM was originally designed to work on large-scale, mainframe computers. R:BASE was developed from the original RIM program.

3.18.33 row

The collection of data defined by a set of columns in a table. Some relational database products refer to a row as a record.

3.18.34 rule

A condition placed upon a column, which R:BASE checks during data entry to ensure that the data meets specific criteria.

3.18.35 runtime

(1) Refers to the actual execution of a program. "At runtime" means while the program is running.

(2) Software that certain applications depend on to run in the computer. The runtime engine must be running in the computer in order for the application to execute. It provides common routines and functions that the applications require, and it typically converts the program, which is in an interim, intermediate language, into machine language. R:BASE Runtime is a separate product available for developers to distribute applications, but with the requirement that the users do will not have access to the development environment.

3.18.36 RX1

The file extension for a database file created using the R:BASE Turbo V-8, R:BASE eXtreme 9.x (64), R:BASE 10.x Enterprise, or R:BASE 11. This file contains the database structure.

3.18.37 RX2

The file extension for a database file created using the R:BASE Turbo V-8, R:BASE eXtreme 9.x (64), R:BASE 10.x Enterprise, or R:BASE 11. This file contains the data for the database.

3.18.38 RX3

The file extension for a database file created using the R:BASE Turbo V-8, R:BASE eXtreme 9.x (64), R:BASE 10.x Enterprise, or R:BASE 11. This file contains the database indexes.

3.18.39 RX4

The file extension for a database file created using the R:BASE Turbo V-8, R:BASE eXtreme 9.x (64), R:BASE 10.x Enterprise, or R:BASE 11. This file contains binary and text large objects.

3.19 S

3.19.1 scale

The number of decimal places in a NUMERIC data type.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.19.2 schema

The structure of the database, which includes the definitions of the tables and their columns.

3.19.3 scratch file

Temporary file created by R:BASE for use in processing commands. See SET SCRATCH.

3.19.4 screen area

A screen area is an area of the form that you define, and its size is determined by the target user's screen resolution.

3.19.5 screen block

Up to 24 lines of text that you can store in a procedure file and display on the screen using the DISPLAY command. Identify screen blocks with the $SCREEN keyword and the screen block name.

3.19.6 screen file

A procedure, ASCII, or snapshot file containing up to 24 lines of text that you can display on the screen using the DISPLAY command. A screen file contains both the information and structure used by the DISPLAY command to display text on the screen.

3.19.7 SELECT clause

See the SELECT clause in the SELECT command.

3.19.8 semicolon (;)

In command syntax, the semicolon is used as a command separator.

Example

The follow commands can be written both ways, the first without semicolons.

```
IF vAnswer = 'YES' THEN
  GOTO vEnd
ENDIF

IF vAnswer = 'YES' THEN ; GOTO vEnd ; ENDIF
```

3.19.9 send to back

Places the current object under another.

3.19.10 send to front

Places the current object on top of another.

3.19.11 server

The computer and disk on a network that holds shared files.

3.19.12 settings

Click this button to specify settings for the current form, report, or label.

3.19.13 single-user

R:BASE system used on a single computer without capabilities for sharing database files.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.19.14 size

(1) For TEXT values, the maximum length of the value.

(2) For NUMERIC values, the precision and scale of the value.

(3) The length of a TEXT column to be indexed without hashing.

3.19.15 sorted order

The organization of data by the values in a specific column or columns before output.

3.19.16 source table

The existing table from which R:BASE retrieves data.

3.19.17 speed menu

A speed menu is a drop down list of options when the right mouse button is clicked. The speed menu contents will vary based upon the current R:BASE module opened, and for the object that is selected when the right click is performed.

3.19.18 SQL

Structured Query Language. A standard language for database definition, display, and manipulation. SQL is a standard methodology established by the American National Standards Institute (ANSI) that is recognized in the database industry as the optimal way to query a table. R:BASE provides the complete ANSI Level 2 SQL, which includes SQL enhancements developed by IBM for its DB2 mainframe database language.

3.19.19 SQL command

Specifies the SQL command to send.

3.19.20 SQLCODE

A system variable that indicates the result of the previous SQL command: 0=success, 100=no rows found, and a negative number indicates failure. Based on the value of *SQLCODE*, the WHENEVER command can be used to call an error-handling routine.

3.19.21 SQLERROR

Indicates that a processing error of any type other than *data-not-found* was detected (*SQLCODE* is less than zero).

3.19.22 standard deviation

Measures the variability or dispersion of a population, a data set, or a probability distribution. Standard deviation can be calculated for a column using the COMPUTE command. Population standard deviation can be calculated for a column using the SELECT command.

3.19.23 startup file

A command file that executes automatically when you load R:BASE, such as RBASE.DAT. See Startup Files.

3.19.24 status bar

The display section at the bottom of the screen used for informational messages about the current window or tool.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.19.25 Stored Procedure

A stored procedure is a collection of R:BASE commands and/or SQL statements that are stored in the database.

3.19.26 STP_RETURN

Contains the value returned by a stored procedure.

The STP_RETURN system variable will contain the value returned by a stored procedure. This return value is set by using the RETURN command. The STP_RETURN variable will be whatever type is needed by the RETURN statement within the stored procedure.

3.19.27 string

A sequence of text characters.

3.19.28 string modifier

Replaces, adds to, or justifies all the characters in a field. In numeric fields, string modifiers identify positive and negative numbers, debits and credits.

3.19.29 structure

The organization of columns, rows, and tables that make up a database.

3.19.30 Structured Query Language

A standard language for database definition, display, and manipulation. SQL is a standard methodology established by the American National Standards Institute (ANSI) that is recognized in the database industry as the optimal way to query a table. R:BASE provides the complete ANSI Level 2 SQL, which includes SQL enhancements developed by IBM for its DB2 mainframe database language.

3.19.31 submenu

A menu that is one level below another menu. You can define as many as 3 levels of menus in an application.

3.19.32 syntax

The structure of a command that incorporates the rules by which R:BASE recognizes the entry as a valid command.

3.19.33 system prompts

In R:BASE versions 6.5++ and lower, there are other prompts that you will run across from time to time when using R:BASE.

Prompt	Description	Exit Method
R>	This is the standard Command Entry prompt. (Exiting this will exit R:BASE)	EXIT or QUIT
B>	Indicates that you have entered a BREAK command but have not reached the end of the looping structure (such as an ENDSW).	In most cases ENDSW. The [Esc] key may work.
I>	This prompt indicates that you are currently within an IF or ELSE body block for	ENDIF

Copyright © 1982-2024 R:BASE Technologies, Inc.

	conditions that are not met.	
L>	This indicates that you are in a LOAD table status.	END
S>	This indicates that you are inside a SWITCH or CASE command block for the non-matching CASEs for a SWITCH but that you have not yet matched a case. Once you have matched a case this is replaced by the B> Prompt.	ENDSW
W>	This indicates that you are inside of a WHILE loop command block.	ENDWH

Examples

The following examples would be typed at the R> Prompt.

The following example shows the I> Prompt.

```
R> IF 'A' = 'B' THEN
I>    WRITE 'TRUE'  *(The I> Prompt indicates this is not
                     being processed and we are waiting
                     for a control structure such as an
                     ELSE or ENDIF)
I> ELSE
R>    WRITE 'FALSE' *(The R> Prompt indicates
commands
                     are being processed. In this case,
                     because A does not equal B. What follows
                     on the next line is the output from
                     the WRITE command.
FALSE
R> ENDIF
```

The following example illustrates the S> and B> Prompts.

```
R> SET VAR vX INT = 1
R> SWITCH (.vX)
S>    CASE 0            *(As with the I> Prompt the S>
                        Prompt indicates a CASE has not
                        matched the value of the vX
                        variable.)
S>       WRITE 'I am zero'
S>       BREAK
S>    CASE 1
R>       WRITE 'I am one'  *(The prompt changes to "R>"

                        because  this is the "active" case.
What
                        follows on the next line is the output
from the
                        WRITE command.)
I am one
R>       BREAK
B>    CASE 2            *(Here is the "B>" where the
SWITCH is
```

Copyright © 1982-2024 R:BASE Technologies, Inc.

```
                    looking for the "ENDSW" command.
This
                    happens because the MATCH has been
                    found and all further commands
                    should be ignored.)
B>      WRITE 'I am two'
B>      BREAK
B> ENDSW
R>                  *(Here we have exited the SWITCH
structure.)
```

The following example illustrates the W> Prompt. Once the ENDWH command is entered many lines of text will be written to the screen. These lines will terminate after one minute.

```
R> SET VAR vLater TIME = (ADDMIN(.#TIME,1))
R> WHILE #TIME < .vLater THEN
W>    WRITE 'Not Yet: ' .#TIME
W> ENDWH
```

The following commands will illustrate the L> Prompt. They also create a new database named TestLPrm which can be deleted.

```
R> CREATE SCHEMA AUTH TestLPrm
R> CREATE TABLE TestLPrm (TestCol INT)
R> LOAD TestLPrm
L> 1            *(The L> Prompt indicates that R:BASE
                is waiting for data to be entered.
L> 2
L> 3
L> END
R>
```

3.19.34 system tables

Tables created by R:BASE to store definitions for specific database items. System table begin with "SYS_".

3.19.35 system variable

A variable that R:BASE creates, such as *#date,* the system variable that holds the current date. See System Variables.

3.20 T

3.20.1 table

Lists of data grouped according to the nature of the data (for example: a list of customers, a list of products, and a list of invoices.) A table is organized by columns and rows.

3.20.2 table lock

In multi-user mode, a command that modifies a table obtains a table lock, preventing simultaneous modification by another user.

3.20.3 tally

To count the number of occurrences of each value in a column.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.20.4 text file

An ASCII file that contains text and no commands.

3.20.5 tier

An area within a region on a form that displays multiple rows of data from the region's table.

3.20.6 tool bar

A bar that contains buttons corresponding to common menu commands.

3.20.7 trigger

Automatically runs a stored procedure when an UPDATE, DELETE, or INSERT command is run on the table. See Triggers.

3.20.8 transaction

A group of commands.

3.20.9 truncate

Make values or text strings shorter to fit into a field of a given length.

3.21 U

3.21.1 Unicode

Unicode is a worldwide character-encoding standard. The system uses Unicode exclusively for character and string manipulation.

3.21.2 union operation

The act of combining the columns and rows of two existing tables into a new table

3.21.3 unique key

A column or set of columns that uniquely identify a row; in other words, each value in a unique-key column is unique. A unique-key constraint prevents duplicate (non-unique) and null values from being entered. The only difference between a unique key and a primary key is that you can define multiple unique keys per table.

3.21.4 user identifier

An assigned name by which a user can access a database.

3.21.5 UTF-8

A variable-width encoding that can represent every character in the Unicode character set. It was designed for backward compatibility with ASCII.

3.22 V

3.22.1 value

Specific number, character, word, or set of words contained in a column or variable.

3.22.2 variable

A symbol that can assume a succession of values, sssigned values, results of an expression, or system variables such as the date or time.

Copyright © 1982-2024 R:BASE Technologies, Inc.

3.22.3 variable form

A form that does not directly refer to a table. Its fields contain variables only. Use variable forms to enter and evaluate data before entering it in tables.

3.22.4 variable object

A field you place on a form, report, or label that corresponds to a variable. In forms, you can enter or display the variable's value; in reports or labels, the variable object displays the variable's value.

3.22.5 variance

Determines the measure of statistical dispersion, averaging the squared distance of its possible values from the expected value (mean). Variance can be calculated for a column using the COMPUTE command. Population variance can be calculated for a column using the SELECT command.

3.22.6 view

A saved query containing columns from tables. A query is basically a SELECT clause that has been saved and can be used again to retrieve data.

3.22.7 virtual storage

A direct access storage device used to store information during processing.

3.23 W

3.23.1 waiting period

Time interval, in seconds, for preventing database resource conflicts in multi-user mode.

3.23.2 watch variable

A variable set to be displayed in order to monitor the variable value for debugging purposes. Watch variables can be monitored during command-file processing in the Trace Debugger, or from the R:BASE main window under "Tools" > "Add/Remove Watch Variables".

3.23.3 WHERE clause

Qualifies or restricts the rows affected by an operation or command. For more information, see WHERE.

3.23.4 wildcards

Characters used to represent one or more characters. The R:BASE wildcard characters are _ (one character) and % (one or more characters). The operating system wildcards are _ (one character) and % (many characters).

3.23.5 Windows

Windows is a series of software operating systems produced by Microsoft. Microsoft first introduced an operating environment named Windows in November 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs).

3.23.6 workstation

A computer connected to a network, providing access to network resources.

Copyright © 1982-2024 R:BASE Technologies, Inc.

Part IV

4 Useful Resources

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

Part V

5 Feedback

Suggestions and Enhancement Requests:

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

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

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

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

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

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

Reporting Bugs:

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

You will need to describe:

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

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

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

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

Copyright © 1982-2024 R:BASE Technologies, Inc.

Index

- # -

- $ -

- % -

- & -

- (-

- * -

- . -

- ; -

- _ -

- + -

- A -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- B -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- C -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- D -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- E -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- F -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- G -

- H -

- I -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- J -

- K -

- L -

- M -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- N -

- O -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- P -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- Q -

- R -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- S -

Copyright © 1982-2024 R:BASE Technologies, Inc.

Copyright © 1982-2024 R:BASE Technologies, Inc.

- T -

- U -

Copyright © 1982-2024 R:BASE Technologies, Inc.

- V -

- W -

- Y -

- Z -

Copyright © 1982-2024 R:BASE Technologies, Inc.

Notes