R:BASE X

and

R:BASE X Enterprise

Custom Table Relationships

R:BASE Technologies, Inc.

Using Custom Table Relationships

Trademarks

Printed: April 2017 in Murrysville, PA

First Edition

1 Introduction

Custom Table Relationships is a new feature within R:BASE X/X Enterprise for multi-table forms and reports, allowing for custom column linking to be defined between tables. By default, R:BASE assumes table relationships are based upon common column names, but the scenario is not always the case for all databases. Table linking relationships can be defined with the following options:

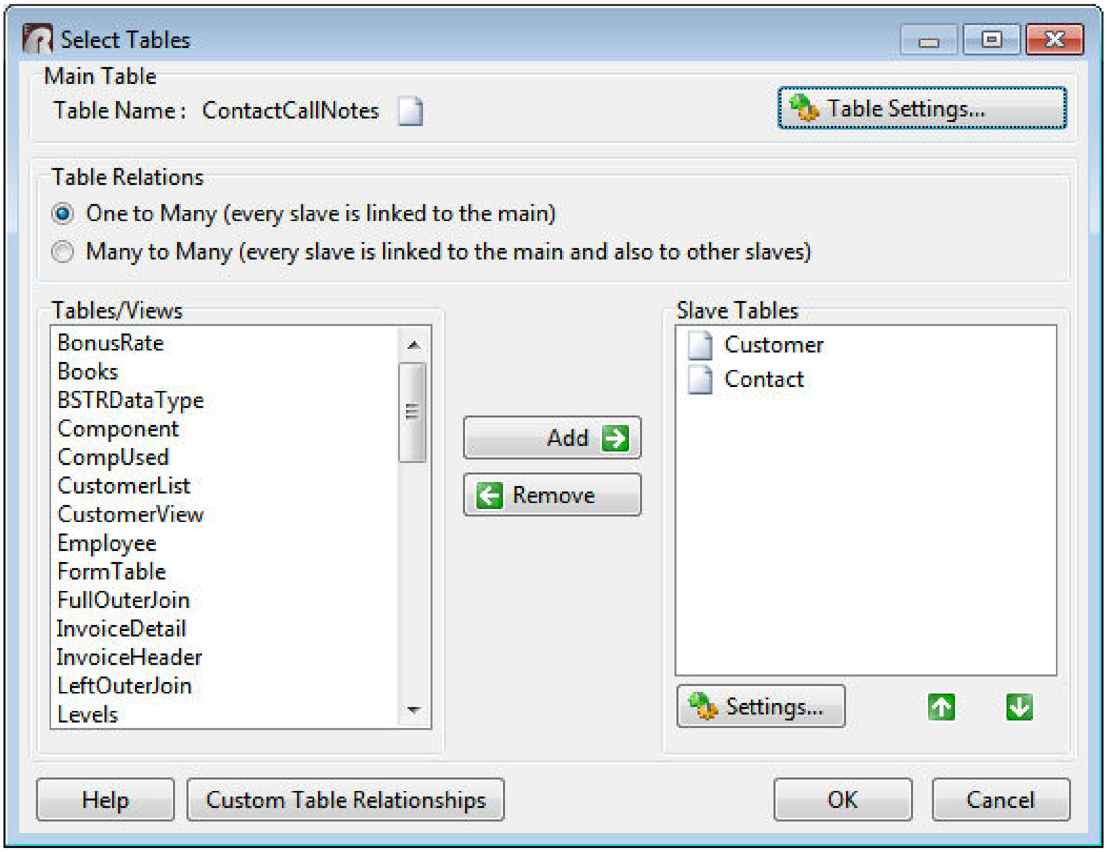
- Common Column Names (default)
- Primary Key/Foreign Key
- Custom Table Relationships

With the introduced custom column linking between tables, the defined relationship is "global" in the scope of the form/report, meaning another form must also be defined with similar linking if the multiple tables and linking relationship exists.

2 Select Tables

Within the Form/Report Designer's "Select Tables" utility, users may add and remove tables, define table relations, alter table row settings, and define table-level Entry/Exit Procedures. The interface is opened by selecting "Tables" > "Add/Remove Tables" from the main menu bar of the designer.

The main table associated with the form/report is displayed within the "Main Table" panel along with a "Table Settings" button. A list of defined tables for the current connected database will be listed within the "Tables/Views" panel, that can also be added to the form/report. Tables added are listed within the "Slave Tables" list.



Main Table

Table Name : ContactCallNotes

Table Relations

● One to Many (every slave is linked to the main)

○ Many to Many (every slave is linked to the main and also to other slaves)

Tables/Views

BonusRate
Books
BSTRDataType
Component
CompUsed
CustomerList
CustomerView
Employee
FormTable
FullOuterJoin
InvoiceDetail
InvoiceHeader
LeftOuterJoin
Levels

Slave Tables

Customer
Contact

Table Relations

The Table Relations panel specifies the form/report logic, which is derived from the common data shared between the tables.

- One to Many - specifies every slave table is linked to the master table. This table 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. With a two-table form, this setting is the default.

- Many to Many - specifies every slave table is linked to the master table and also to the other slave tables. With a two-table form, this table relationship specifies that Table 1 can contain many rows for any given row in Table 2, and vice versa. With more than two tables assigned to the form, this relationship can also mean that for any given row in Table 1, there are many rows in Table 2. And, for any given row in Table 3, there are many rows in Table 2. Usually, three tables are used in this relationship with two of the tables using a common column in relation to a third. These two tables do not link directly to each other; instead, each of these link to a third table in a One to Many relationship by a common key column with values that match. With more than two tables assigned to the form, this option is the appropriate setting.

Slave Tables

Slave tables are additional tables added to forms/reports. Once added, the table is listed within the "Slave Tables" list. A "Settings..." button is available for any selected slave table. Using the green up/down arrows, slave tables should be arranged for order of processing, which is significant. An example of a many-to-many relationship with four tables is: Slave1 links to Master. Slave2 links to Master, then Slave1. Slave3 links to Master, then Slave1, then Slave2.

Dataset Behavior

The contents of the datasets displayed upon a form/report will vary based on the selected relationships. Many-to-many relationship amounts to fewer result sets in slave tables. Consider the following example:

Tables

```
Main..: Field1, Field2, Field3
Slave1: Field1, Field2, Field3
Slave2: Field1, Field2, Field3
```

Field Relationships

```
Main.Field1   -> Slave1.Field1
Main.Field1   -> Slave2.Field1
Slave1.Field2 -> Slave2.Field2
```

One To Many Scenario

```
Main..: Field1, Field2, Field3
        |
Slave1: +--> Field1, Field2, Field3
        |
Slave2: +--> Field1, Field2, Field3
```

Many To Many Scenario

```
Main..: Field1, Field2, Field3
        |
Slave1: +--> Field1, Field2, Field3
        |            |
Slave2: +--> Field1, +--> Field2, Field3
```

- In a one-to-many relationship, the relationship between Slave1 and Slave2 (slave-to-slave) is discarded. Only the equality of linked fields between the master and a specific slave is implemented.
  ```
  Slave1 WHERE: (Main.Field1 = Slave1.Field1)
  Slave2 WHERE: (Main.Field1 = Slave2.Field1)
  ```

- In a many-to-many relationship, all table links are implemented.
  ```
  Slave1 WHERE: (Main.Field1 = Slave1.Field1)
  Slave2 WHERE: (Main.Field1 = Slave2.Field1) AND (Slave1.Field2 = Slave2.Field2)
  ```

- The order of tables is important. In the sample above the order is (1) Main, (2) Slave1, (3) Slave2

- If a record pointer is moved in a table all tables lower in the "hierarchy" are refreshed using an updated WHERE clause (using new row values in the navigated table). This means if Main is navigated then tables Slave1 and Slave2 are refreshed. If Slave1 is navigated Slave2 is refreshed. If Slave2 is navigated no other table is refreshed.

3 Custom Table Relationships

Custom table relationships are available for multi-table forms/reports to set specific column links between tables. The interface provides a natural method to click and drag a column name to another column name for creating links. This is where table relationships are defined, regardless of order (in the slave tables list).

For a "Main Table", select the "Custom Table Relationships" button within the Select Tables window to define custom table relationships.

For a "Slave Table", click to highlight the specific table in the Slave Table list, and click the "Table Settings" button. Within the Table Settings window, notice the default table relationship is Common Field Names. Choose the "Use Links in Custom Table Relationships" option, then the "Custom Table Relationships" button. By selecting to "Use Links In Custom Table Relationships", the links in the Custom Table Relationships will be used to link with the main table and other slave tables higher in the slave list. Every slave table has its own relationship settings.

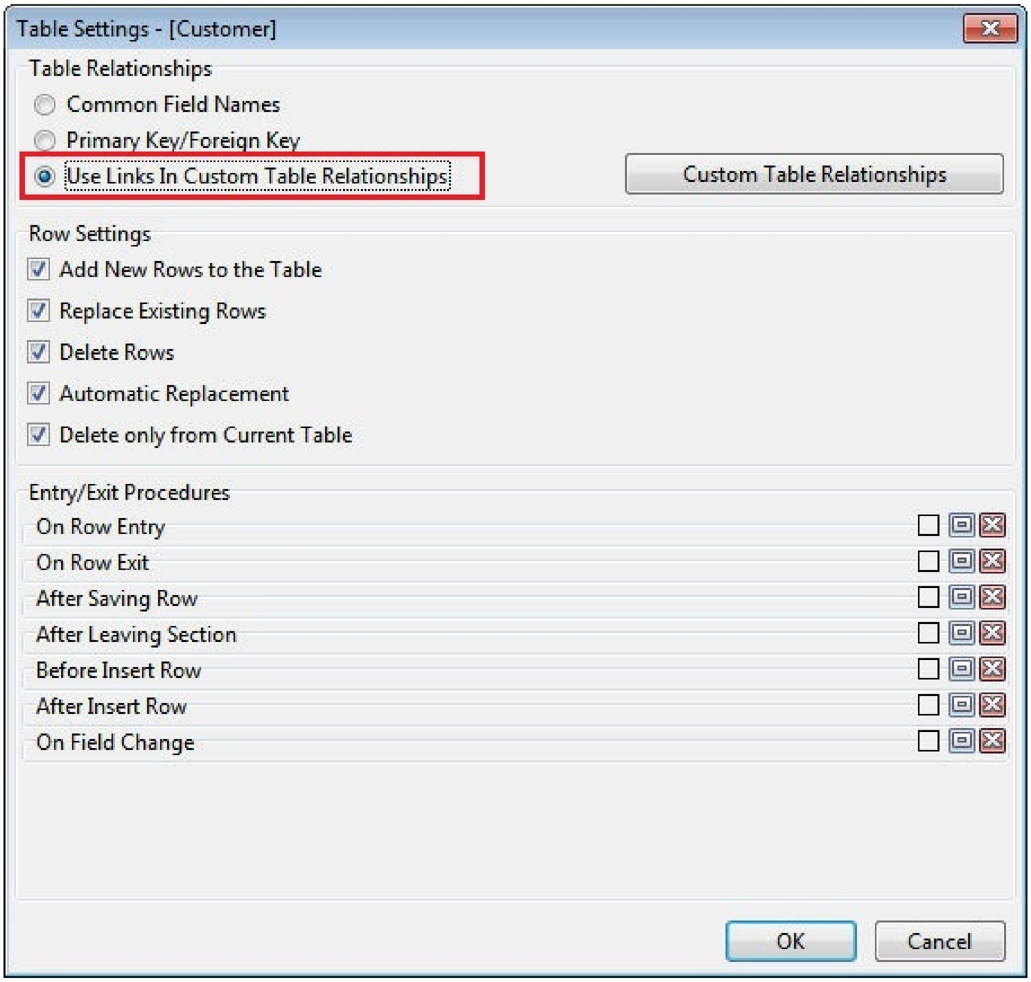


Table Settings - [Customer]

Table Relationships

○ Common Field Names

○ Primary Key/Foreign Key

● Use Links In Custom Table Relationships

Row Settings

☑ Add New Rows to the Table

☑ Replace Existing Rows

☑ Delete Rows

☑ Automatic Replacement

☑ Delete only from Current Table

Entry/Exit Procedures

On Row Entry □

On Row Exit □

After Saving Row □

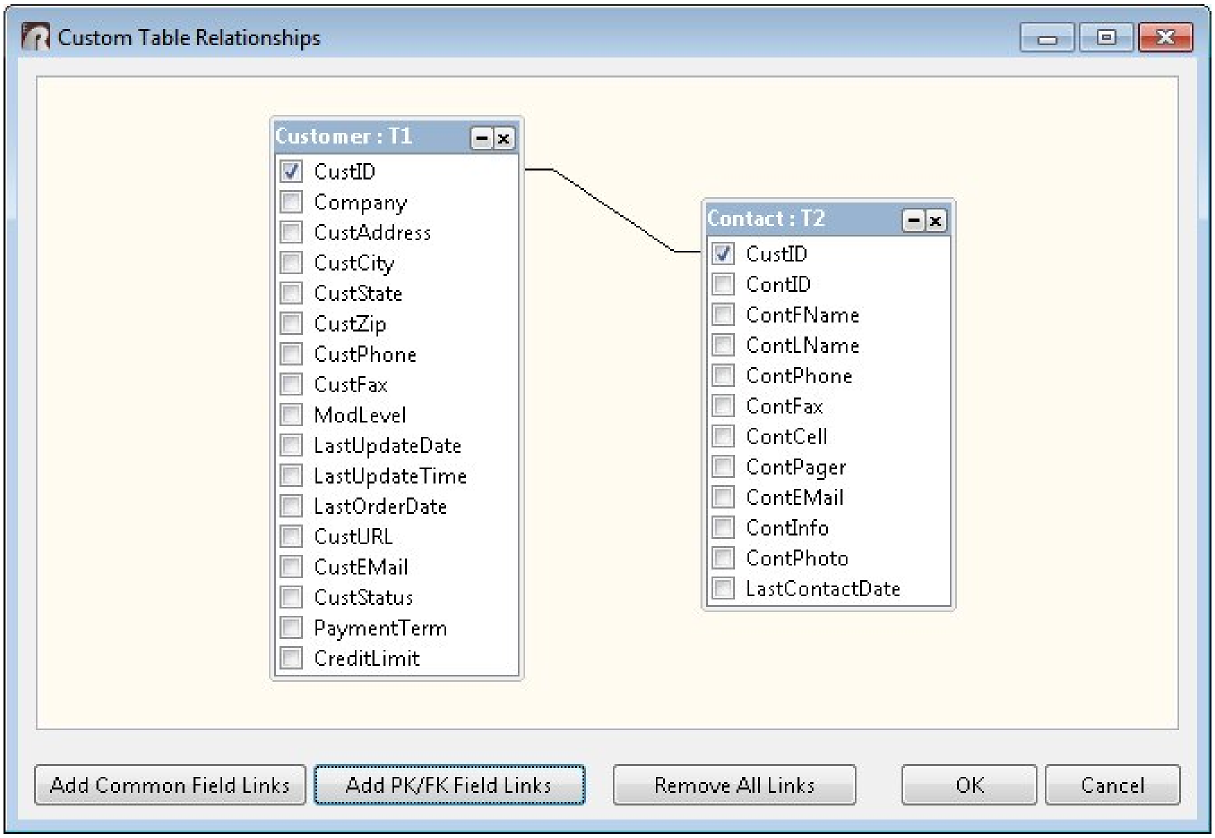
After Leaving Section □

Before Insert Row □

After Insert Row □

On Field Change □

Buttons are available to provide automatic field links, and to remove defined links.



Defining Custom Relationships

Links between tables can be customized if common field names or primary/foreign key relationships were not used in the database design, or if some other purposeful convention is used between tables. Custom links between the tables would be based upon the columns with the one to many or many to many relationships.

To create a custom table relationship, first remove existing links, then link fields using the mouse cursor to drag a column name from one table to the linking column in the other table. A link can also be removed by right clicking on the linking line, and selecting "Unlink" from the menu. To review the properties of the link, right click on the linking line, and select "Link Relationship" from the menu.

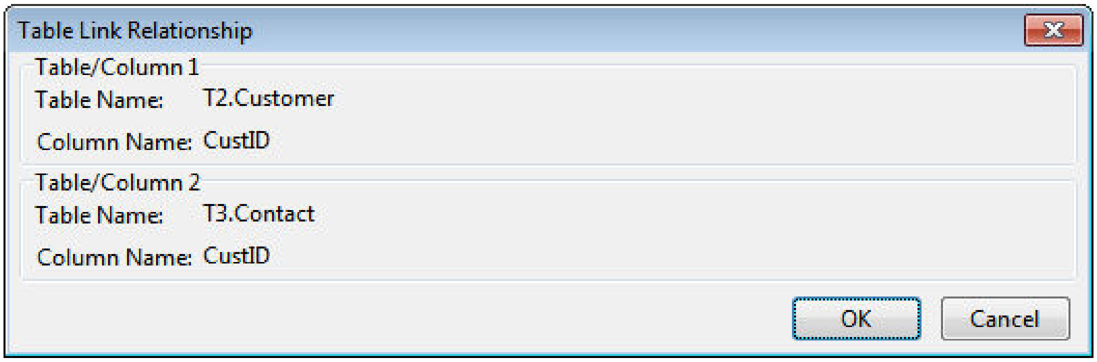


Table Link Relationship

Table/Column 1

Table Name: T2.Customer

Column Name: CustID

Table/Column 2

Table Name: T3.Contact

Column Name: CustID

When using many to many relationships with several slave tables, it is important to note that every slave table maintains a relationship with the main table **and** the other slave tables, and the order of the slave tables is significant.

For example, SlaveTable1 links to the Main table. SlaveTable2 links to the Main table, then SlaveTable1. SlaveTable3 links to the Main table, then SlaveTable1, then SlaveTable2. This is how many to many table relationships are defined regardless of order in the slave tables list. Using custom table relationships, the links can defined in a more intuitive method.

4 Examples

Example 01. - Unintentional Common Columns in a Transaction Form

In a legacy database migration from DOS, a "transaction" form contains an extra set of common columns which disrupts the display of slave table data when running the form. In the below tables, fields, and relationships, the Invoice_Number field is the expected common column link between the Invoice_Header and Invoice_Transaction tables. However, a LastUpdated column is used in both tables as well, only containing different and unreferenced data, one being the last date the invoice header information was updated, and the other being the last date when an invoice transaction item was updated.

Tables

```
Invoice_Header.....: Invoice_Number, LastUpdated
Invoice_Transaction: Invoice_Number, LastUpdated
```

Field Relationships (common column)

```
Invoice_Header.Invoice_Number ->  Invoice_Transaction.Invoice_Number
Invoice_Header.LastUpdated    ->  Invoice_Transaction.LastUpdated
```

The solution is to create a custom table relationships for the form, specifically for the Invoice_Number field. The defined custom table relationship would avoid having to change one of the LastUpdated column names, which would require a review of the table definition, application code, and form/report variables used throughout other forms and reports, where the column may have been used in other places.

Steps:

1. When reviewing the form tables, the "Table Relations" should be set to "One to Many", as Invoice_Header can contain only one row for any given row in Invoice_Transaction, but Invoice_Transaction can contain many rows for any given row in Invoice_Header.

2. For the main table, Invoice_Header, select the "Custom Table Relationships" button from the "Select Tables" window. Within the Custom Table Relationships interface, remove the existing links with the "Remove All Links" button. Then, link the table fields using the mouse cursor to drag the Invoice_Number column name from Invoice_Header to the Invoice_Number column within Invoice_Transaction. Click the OK button to save the link.

3. For the Slave table (Invoice_Transaction), the Custom Table Relationships must also be assigned to use "Invoice_Number" only. Within the Select Tables dialog, click to highlight the Invoice_Transaction table in the Slave Table list, and click the "Table Settings" button. Within the Table Settings window, choose the "Use Links in Custom Table Relationships" option, then the "Custom Table Relationships" button. As done previously, remove the existing links with the "Remove All Links" button, then link the table fields using the mouse cursor to drag the Invoice_Number column name from Invoice_Header to the Invoice_Number column within Invoice_Transaction. Click the OK button to save this link.

4. Continue pressing the OK button for the windows to save changes and return to the Form Designer, then save the form.

After the above changes, the slave table data will appear when running the form.

Example 02 - Common Columns in a Sub Report

In the development of a Sub Report within a new report, using Customer and Contact relationship extra sets of common columns disrupts the display of slave table data from displaying on the Sub Report. In the below tables, fields, and relationships, the ContactID field is the expected common column link between the Customer and Contact tables. However, the Address, City, State, ZipCode, and PhoneNumber columns are used in both tables as well, only containing different and unreferenced data, one set being the address and phone number information for Customers, and the other set being address and phone number information for Contacts, which technically can be different.

Tables

```
Customer.: ContactID, Address, City, State, ZipCode, PhoneNumber
Contact..: ContactID, Address, City, State, ZipCode, PhoneNumber
```

Field Relationships (common columns)

```
Customer.ContactID   ->  Contact.ContactID
Customer.Address     ->  Contact.Address
Customer.City        ->  Contact.City
Customer.State       ->  Contact.State
Customer.ZipCode     ->  Contact.ZipCode
Customer.PhoneNumber ->  Contact.PhoneNumber
```

The solution is to create a custom table relationships for the report, specifically for the ContactID field. The defined custom table relationship would avoid having to change all of the Address, City, State, ZipCode, and PhoneNumber column names for either table, which would require a review of the table definition, application code, and form/report variables used throughout other forms and reports, where the column may have been used in other places.

Steps:

1. When reviewing the report tables, the "Table Relations" should be set to "One to Many", as a Customer can contain only one row for any given row in Contact, but Contact can contain many rows for any given row in Customer.

2. For the main table, Customer, select the "Custom Table Relationships" button from the "Select Tables" window. Within the Custom Table Relationships interface, remove the existing links with the "Remove All Links" button. Then, link the table fields using the mouse cursor to drag the ContactID column name from Customer to the ContactID column within Contact. Click the OK button to save the link.

3. For the Slave table (Contact), the Custom Table Relationships must also be assigned to use "ContactID" only. Within the Select Tables dialog, click to highlight the Contact table in the Slave Table list, and click the "Table Settings" button. Within the Table Settings window, choose the "Use Links in Custom Table Relationships" option, then the "Custom Table Relationships" button. As done previously, remove the existing links with the "Remove All Links" button, then link the table fields using the mouse cursor to drag the ContactID column name from Customer to the ContactID column within Contact. Click the OK button to save this link.

4. Continue pressing the OK button for the windows to save changes and return to the Report Designer, then save the report.

After the above changes, the Sub Report table data will appear when running the report.

5 Useful Resources

. R:BASE Home Page:	http://www.rbase.com
. R:BASE X Home Page:	http://www.rbasex.com
. Up-to-Date R:BASE Updates:	http://www.rupdates.com
. Sample Applications:	http://www.rbasecommunity.com
. General R:BASE Syntax:	http://www.rsyntax.com
. Technical Documents - From The Edge:	http://www.razzak.com/fte
. More Sample Applications:	http://www.razzak.com/sampleapplications
. Education and Training:	http://www.rbaseuniversity.com
. Upcoming Events:	http://www.rbase.com/events
. R:BASE Beginners Tutorial:	http://www.rtutorial.com

Notes