

Understanding DataEase

Tables, Forms and [F10]

Graham Smith
PLM Consulting, Inc.
gsmith@plmconsulting.com
July 2006

One of the biggest advantages of moving from DFD to DFW is the fact that the data Table and the Form no longer need to be interlocked. This means that it is possible to have multiple forms on the same table and that not all the fields in the table have to be on any one form. The downside to this is that [F10] can no longer be counted on to work as expected for the very simple reason that there is no way to tell it which form to open. To explain this better, we need to look at the difference between a table and a form and what [F10] does and how it works.

Roughly speaking, a table is the structure that holds the data and the form is the user interface for that data. When you create a form in DFD, you are also creating the table and the two things cannot be separated. You cannot move a field in a form without it changing the table layout as well and you cannot remove a field without removing it from the table.

In DFW, you also use a form to create and manage a table. This is referred to as the table defining or table owning form (TOF). But unlike DFD, you can create additional forms on the same table in DFW. These are referred to as table using forms (TUF).

Any table can be related to one or more additional tables. In addition, there can be multiple relationships between the same two tables. To sort all this out, it is common practice to name the relationships in some way that will represent the link in an understandable manner. When [F10] is pressed in a DFD form, it will display the list of relationship names that this table has. When you choose one, it will move you to that form. But in DFW, there can be more than one form so the best it can do is to take you to whatever form has the same name as the table. This immediately creates a real issue for the [F10] function for some very complicated reasons that the original designers failed to take into consideration.

When a table is first created in DFW, it is going to have the same name as the form that is used to create it (the table and the TOF always start with the same name). In some cases, the story will end right there because no other form is ever going to be needed. But years ago, developers found that there was a distinct advantage to having a TUF that they could edit and rearrange as much as they wanted without doing anything that changed the table. This was of particular concern in the early days when the product was less stable than it is now and frequent changes could result in a trashed TOF and table. This was also when developers first discovered that they were going to have to find a replacement for [F10].

At about this same time, a group of developers started looking at the idea that displaying an unsorted list of relationship names to an end user was far from the most efficient way of allowing them to move about. They also realized that by eliminating [F10] they could also keep the users from seeing relationships they did not want them to see. Thus was born and grew "The search for a better way to [F10]."

There are three main solutions used in this application:

- A button on a subform record to open that record (a “drill down” button)
- Buttons on a form to open a related form
- A menu list of related forms

Below are examples of each:

Item	Date of Hire	Term Date	Term Code	Group Code	Location
1	08/26/1991	06/11/1999	RED	3	Woody Springs
2	08/26/2002				

The button to the left of the record will open a form and display that specific record. Unlike DFD, no records other than the one opened can be seen from the newly opened subform.

To move to a related form and display all the records, a button may be placed on the subform itself, but not on a record like this:

Item	Date of Hire	Term Date	Term Code	Group Code	Location

The third way to allow access to related records is by way of the menus.

The screenshot shows a software interface with a menu titled "User Defined Menu". The main window is titled "People" and contains various input fields for employee information such as "Emp ID", "Original Hire", "Age", "Retirement Date", "Pending Disability", "Job Code", "Nickname", "Marital Status", "SSN", "Gender", "Orientation Date Sent", "Termination Date", "Eligible for Partial", "Eligible to Retire", "Eligible to Diversify", "Settled", "Rehired", and "TEFRA 242(b) On File?". A "Related Forms" menu is open, listing several options: "Address Information", "Account Balance", "Diversify Calc.", "Partial Calc.", "QDRO Recipient", "PS Earnings", "Retirement (Current)", "Retirements (All)", "Retirement Calc.", "Vested Termination (Current)", "Vested Terminations (All)", and "All Related Forms".

Choosing one of the menu items will open the related form and display all the related records. It

is important to note that opening a form this way sets a filter such that only the related records can be seen.

What each of these three techniques have in common is that they allow the developer to specify not only what relationship is going to be opened, but what form to use as well.

Creating a Drill-Down Button on a Subform Record

Using this technique can cause a lot of head-scratching until you realize that you are placing a button on the subform record, thus any references you make in it to relationships refer to the subform table, and not the mainform table. Thus, to open HiringHistory, you need a relationship between HiringHistory and itself based on the unique field. You will need to give this an optional relationship name and I use `_NA` and `_Self` for the two sides (`_NA` is short for not applicable or don't use). The button action then is

```
FormOpenRelated(_Self, _HiringHistory)
```

This also brings up an unrelated topic which I will only mention here. Since the `FormOpenRelated` button action on a subform refers to relationships to the subform itself, you can actually open a related table in a way that is impossible in DFD. For example, if you had a form on `OrderHeader` with `OrderDetail` as a subform, you could place a button on the `OrderDetail` subform that would open the `Products` table that `OrderDetail` looks up from. A pretty clever trick in some cases.