Developer's Diary: D

Freed THOU

Adrian Jones offers a pot-pourri of snippets for developing in DataEase for Windows

Token ised Relation ships

I have found an important difference between the way that DOS and DfW handle "pre-defined" relationships used in DQLs, that may give you surprising results.

In a DOS DQL, the details of a predefined relationship are stored in the actual DQL. So if you have a join between Customer and Invoice based on CustomerID being equal, and use that relationship by name in a DQL, then the match field details will be stored in the DQL (though you won't actually see this).

If you later modify the System Relationships record, say to add some new match fields, your DQL will continue to run as though the old relationship was in place. This is the case even if you delete the Relationships record. In other words, a DOS DQL knows nothing about changes in the System Relationships table. Until, that is, you re-compile the DQL by opening it up and hitting F2.

Not so in DfW! Usually, any change you make to the predefined relationships are reflected when you run the DQL. In other words, DfW checks the relationships each time, and does not store the match fields within a DQL.

The exception appears to be when you use a predefined relationship, but then rename it in the DQL. In these circumstances, any changes to the Relationship record will not be reflected until you recompile the DQL. That is, DfW behaves the same as DOS.

But — incredibly! — if you name an ad-hoc relationship the same as a pre-defined one, changes to the predefined are reflected in the DQL.

So, with a predefined relationship between Customer and Invoice called SeeInvoices:

```
For Customer ;
list records
  count of Invoice
named "SeeInvoices" .
```

will give you the count of Invoices for each customer, despite there apparently being no match fields.

This is also not the case in DOS.

Best Practise

In an ideal world, we'd all get our database structure completely perfect before we went anywhere near a DQL. But we'd also pay our taxes on time, never run up overdrafts or speed up for amber lights.

In reality, and especially in more complicated applications where relationships are used to perform a number of clever tricks, there is always the chance that we will change a relationship definition somewhere.

So it seems best to think that you want a DQL to function as you wrote it, and not to change later. Therefore:

1. Always name your relationships.
   Otherwise there is no clear way of distinguishing between a table and a relationship.
2. Don't use pre-defined relationships in DQLs. Instead, name them and restate them.
3. Don't use the same name in your DQL for the relationship as a pre-defined one!

It is practise to do this last one in both DOS and DfW, to 'keep the tigers away', and you could adopt a simple habit of, say, naming the relationship "AdHocRelName".

No Name To Blame

Now, we all know that we should name our relationships. If we don't give a relationship a name, in DOS or Windows, DataEase will do it for us, using the name of the table instead.

In DOS, it is possible, because of the interactive menu definition, to distinguish between Invoice the table and Invoice the relationship. In DfW, it is not — even though you can pick the same name from the list of tables or relationships, what you enter in the text editor is text; there does not appear to be any background hidden distinction.

And even with DOS, if you recompile the DQL, it will assume that you meant Invoice the relationship, not the table.

All this underlines that most-repeated piece of DataEase advice — name your relationships!

DataEase for Windows

om of JGHT

Memo on Memos

DfW replaced the DOS long:text fields with the memo field. Or did it?

DOS users wanting to store text about customers or products will have used a series of long:text fields to give them limited but useful word-processing functionality, most notably word-wrapping.

Nosing through their copy of DfW, they will have discovered that they now have memo fields, which allow up to 4,000 characters. Or, if they migrated from DOS, they may have noticed that the long:texts had been converted to memos.

In fact, when you create a memo field, you are really creating a set of text fields on the table. If you ask for a memo field 1,000 letters in total, you will have made three fields each of 255 characters, and a fourth of 235 to make up the total length of the four fields to 1,000 characters.

If you named this field "Memo", the first field is called Memo, the next, Memo1, then Memo2, etc. Except that you don't get to see this ever, unless you go in through the back door by opening up your application with DOS 5.

Now, one Dialogue reader was experiencing problems with memos. He was importing records from one workstation to another and found that, if the record was new, DfW imported correctly. However, if it already existed, he sometimes got a jumble of old and new bits of text.

The reason is quite simple. Let's imagine that an original record filled more or less the entire 1,000 characters in its memo field. Thus it had data values in the fields MemoField, MemoField1, MemoField2 and MemoField3.

The other workstation updated this record with a shorted memo, say only filling MemoField1. Or it had a new description in there that filled up the whole four fields. Either way, when the data got imported, DfW only changed the value in MemoField. In other words, only the first 255 characters were changed. Hence you can end up with imported values that are a mixture of old and new.

A Solution

Whilst DfW won't display anywhere — including it the export dialog — that there are four fields for what appears to be one, you can still type in the 'background' field names and it will understand what you want.

So, my first thought was to write an export DQL that replaces the single MemoField with the four underlying ones. Thus:

```
for Customer ;
  list records
    CustomerID ;
    MemoField ;
    MemoField1 ;
    MemoField2 ;
    MemoField3 .
end

export to "c:\temp\memo.dat" .
.items
@f[1,1]~ @f[1,2]~ @f[1,3]~
@f[1,4]
.end
```

This works beautifully except for the minor detail that it will crash your computer!

However, by assigning temporary variables, each of 255 characters in length, to the various sections of MemoField, we have a solution:

```
define temp "tMemExp1" text
       255 .
define temp "tMemExp2" text
       255 .
define temp "tMemExp3" text
       255 .
define temp "tMemExp4" text
       255 .

for Customer ;
  tMemExp1 := MemoField .
  tMemExp2 := MemoField1 .
  tMemExp3 := MemoField2 .
  tMemExp4 := MemoField3 .

  list records
    CustomerID ;
    tMemExp1 ;
    tMemExp2 ;
    tMemExp3 ;
    tMemExp4 .
end

export to "c:\temp\memo.dat" .
.items
@f[1,1]~ @f[1,2]~ @f[1,3]~
@f[1,4]
.end
```

It doesn't matter that the final temporary variable is also 255 characters — any spaces will be truncated anyway.

You can now import the records on the other workstation.

Incidentally, what is a bit confusing here is that when you export the records by simply including MemoField in the DQL or the via the export dialog, the entire string is exported (check this by opening the export file in a text editor), so you get the impression that everything will work properly. Unfortunately, it's the import side that is the problem, as this only imports the values into the first field of the memo set, and doesn't change the values in the other fields.

Take Exception

You don't have to go far beyond tearing off the cellophane wrapping to "enjoy" the company of DfW's exception handler. This was added by frustrated DfW programmers in the USA, desperate to track down the circumstances when DfW crashes.

The idea was you'd save the report, write down what you were doing when it went wrong, and get the details to the development team, who'd give it their immediate attention.

Of course, that team is no more. And probably the information that the exception handler gives is as baffling to them as it is to the rest of us.

Well, in working through the DfW code, Pete Tabord has discovered that the handler was only supposed to work with Windows 3.11 anyway. And that it itself is very possibly the cause of some GPFs.

So the best advice is — turn it off! Add a variable to your autoexec.bat file that says:

```
set DE_EXCEPTION_DISABLE=OFF
```

Every time you load DfW, you'll get an error message telling you that the handler is disabled — just tell your users to click okay.

It's hard to pin down exactly what causes DfW GPFs. sometimes it seems to be down to just the workstation. I've even had problems with specific brands of computers. In one particular instance, DfW would crash around about the third time someone ran a procedure. With the exception handler off, the 'an error has occurred ... close/ignore' dialog might still appear, but three clicks of the ignore button latter, and the same procedure continued to run as planned. Okay, I would still prefer crash-free running, but this was at least a workable compromise.

In the just released DfW 5.14 the handler will be turned off automatically if it detects that you are running anything other than Windows 3.11.

Undeleting with JADA

Have you ever deleted a record you didn't mean to? Worse, did you delete both the main record and all its related subrecords?

Well, I just did. It's this sticky keyboard; the down arrow didn't down arrow enough. But enough of these excuses.

In DOS, we might work out the record number of a deleted record, and restore it using Ctrl-F3. No such option exists in DfW, but we can use the DOS third-party add-in JADA Tools to access the tables in our application and recover the record. Yes — JADA will read DfW tables, including the RDRR.

JADA has three options for this, found on the 'Deleted Records' submenu of the main menu option 'Troubleshooting'. "Undelete all records" first displays a list of the forms/tables in your system. Select your table and all deleted records are restored. "List undeleted record numbers" produces a list to send to your printer, disk or screen of the 'record number' of each deleted record. (I've put this in quotes because the record number does not physically exist on each record. It's merely the position in the file of the record.) Finally, 'New Form Using Deleted Records' creates a copy of the table from which you are restoring records and populates it with all deleted records, so you can selectively copy the ones you want across.

The latter option is preferable if you haven't reorganised this table for a while and it contains a number of records that you don't want restored. But anyway, I headed for option 1 to restore the lot. I then

Summing Up *Wrong*

A glitch can ruin the effect of summary variables displaying on-screen in form and live report documents.

If you are displaying more than one record to which a summary variable is applied and the user clicks on the next record button/F3 (as well as previous record, and by clicking on the up or down arrows on an appropriate scrollbar), the wrong summary values will probably be displayed. DfW seems to get stuck on the value for a previous record.

This does not apply when the user is moving down to the next page of records, or jumps to the first and last records.

As a workaround, either only display one record at a time, or restrict how the user can move between records. To do the latter, you will first need to remove the toolbar, because you can neither selectively disable any of the buttons, nor customise any button action.

Then remove the 'next record' and 'previous record' commands from the goto menu, and add a couple of buttons to the document that take the user to the next and previous pages (to make up for the lack of a toolbar).

Unfortunately, you'll still be stuck with the scrollbar's behaviour, so (if this is a subform) either don't display the scrollbar, or explain to the user that they should only click on the ribbon, not on the up/down arrows, if they want accurate, on-screen summary values.

But as long as they move between records a 'page' at a time, the results should be correct.

A virtual field derived as a "sum of RelatedTable" will work no matter how the user moves between records.

returned to DataEase and was surprised to see that, whilst the records were there in the main table and in the table that was used as a subform, I couldn't get the main form to display any of the previously deleted subform records.

The relationship between the main and subform was, as you'd expect, based on an indexed field. I realised that JADA probably hadn't done anything with the indices when it restored the records. When I reorganised the table, the subrecords displayed against the corresponding main record.

So, the lesson is: update the indices after restoring deleted records with JADA.

APP Objects

Last issue we peaked at writing reports over the system table Application Objects. I said that this table is supposed to be hidden from us mortals, but for the time being, the key to it is still left under the mat. So, before Pete Tabord changes the locks, let's break in again!

The .DIW file routine in DfW is the equivalent of the .DIN file in DOS, and is used to maintain 'remote' systems — i.e. ones that for reasons of distance or time, you won't be able to visit to make the modifications directly. Unfortunately, the .DIW file doesn't work very well, and until this is fixed (it's been promised for the next patch after 5.14), you can't reliably use it.

This idea is an extension of what I talked about last issue, but has certainly helped me make changes to remote systems. It simply writes a DOS batch file that I can use to copy all the documents I have changed since my last site visit to a floppy disk, ready to re-install at the client site.

Moving on from the DQL example last issue, this simply says:

```
For Application Objects with
date ( midc ( Description ,
1 , 2 ) ,
     midc(Description,4,2),
     midc(Description,7,2))
>= data-entry DateSinceModified;
  list records
    jointext ( "copy " ,
jointext(File Name," a:")).
end

export to "maintain.bat" .
.items
@f[1,1]
.end
```

I can then run this batch file from a DOS prompt. And when the .DIW route has been given the all-clear, we can modify the whole procedure to write the .DIW file as well as copy the appropriate files to disk.

In the meantime, this can then be used in conjunction with the printout from last issue to add or update those forms.
A few notes here.

1. Export all the data from your relationships table as well and include this on disk.
2. Assume that all changes you've made to forms that define tables have been to the table themselves, and don't re-install these. Instead, write down such changes and make them manually.
3. You will have to first delete those documents that already exist and have since been modified. As long as they don't define forms, you can delete them okay.
4. Don't update directly onto the network. Copy the database to a local drive, make the changes, then copy back up again.

Float On

The DfW toolbar is generally something you can't customise. It is either on, or it's off. When it's on, you get what you're given in turns of buttons and what they do.

The only thing you can change is where it appears on screen by adding a setting to the DFW.INI file. The variable is called Location, and a value of 1 positions the toolbar at the top of the screen, 2 at the bottom, 3 at the left-hand side, and 4 to the right. Thus:

```
Location=2
```

would give you a toolbar at the bottom of the screen, just above the status line.

This was documented in the 5.13 README file, and duly reproduced in a past Dialogue. What is not documented anywhere as far as I know, is that you can right-click on any non-button part of the toolbar and drag it to wherever you like. This can include one of the edges of the screen, but can also be freely floating right next to where you are working.

This can be especially useful when, as developer, you are creating the bits and pieces that make up your application.

To tidy the toolbar away, simply right-click again and drag it in the direction of one of the edges of the DfW session. When you see the outline of the toolbar change to a long, thin rectangle, you can let the mouse go.

The toolbar will be back in its normal place the next time you start DfW.