Simply Higher

An in-depth primer to help you create your first One Level Higher form, and understand what you are doing!

Adrian Jones MBE, BA (Hons), May 2006 www.n10net.com

Let's start with the most simple implementation of the One Level Higher concept. In it I will cover all the basic things you need to do, to save having to repeat those details in the later variations. I'll cover the basics in detail, but you can cut to the chase by looking at the 'five minute checklist' at the end of this article.

The Top Table

The concept uses a top, parent table, from which other related tables are then 'hung'. Any table can act as the parent. I use one I call Looper, so named because when I first started using it, I wanted to 'loop' x number of times through a 'for' statement in a DQL. Looper is really a set of ordered integers, or ordinal numbers, stored in a field called LooperNo.

Integral to a flexible implementation of One Level Higher is the use of the custom functions named get/setglobal. These functions refer to their 'contents' via a number, so we can match the value in the global with the value of the Looper record.

Currently, the simplest way to persist a value throughout a DataEase For Windows application session is using these functions. Were Sapphire to change, say, DQL global variables so we could refer to them in forms, or were a third-party CDF writer to build a library that let us refer to an in-memory value by name, then other options would open up.

To simplify the discussion, I will refer to these as global values. Please don't confuse them with DQL globals…

Looper has a virtual field vMatch that picks up the global value, and is used as the the match field in relationships to the 'true' table records we want. vMatch is derived:

```
If ( LooperNo <= 255 , getglobal ( LooperNo ) , blank )
```

The get/setglobal library affords us 255 variables, which should be sufficient for most application sessions. The 'if' statement simply helps avoid the GPF we will get if we try to access nonexistent values at location 256 and beyond.

I will discuss Looper in depth in a later article.

Relationship

To display your target as a subform of the parent, you'll need a relationship between the two.

With Looper, each relationship will match vMatch. I always put the one, parenting side of a relationship on the left hand, and the target table on the right. See my article on Relationships for more.

Which target field you match depends on what you are trying to achieve. To display the one and only record for a given value, use that table's ID. If you target anything other than the identifier, then that side is by definition the many side.

This might contradict some things you've been taught, but because vMatch is a long-enough text field, it will match ANY field type, with the exception of a floating point number. Extended or standard date, time, any number other than float, choice, numeric string; it doesn't matter. See the document ViewLooperMatches in my demonstration application if you need proof.

Name your relationship appropriately. I suggest if you are matching on the ID, you call it See[Table], otherwise name it See[Table]s[MatchField], so SeeCustomersDOB. Of course, you will have to work within the relationship name limit (which appears to still be 20 characters in version 7).

The Looper side should always be named NA for 'Not Applicable'; it makes no sense to navigate from a child record to its artificial parent, and because this is matching on a virtual, DfW may GPF if you do.

It won't crash going from the parent to the child, though, because it is a trivial process to retrieve the value of vMatch for each record.

Fields

You do not need any fields from Looper in your final form. However, when you first select the table, you will have to choose at least one, otherwise DfW adds all fields from the table.

Navigating and Deleting

Basically, there is no 'next' record, and the parent form should be filtered so that it is fixed on the one and only record you want. Therefore any record navigation options should be removed. Ditto clear form, as it will only confuse the screen.

The delete records option must be removed as well (from toolbar and menus). If you allow deleting, control that from the subform, which means using CDFs to get round the generally confusing delete dialog.

The Five Minute Checklist

1. Create a parenting table with a virtual to pick up a given array value
2. Create relationships to true target (i.e. the table you are really interested in)
3. Build new forms starting with the parenting table, and your target as the subform.
4. No fields from the parent need appear on the form.
5. If 1:1 relationship, remove scrollbars from the child.
6. Set document properties to display the first record.
7. Set a filter on the parent to match the given ID, e.g. LooperNo = 1
8. Remove delete records and clear form options from the menus and toolbar

---

§ ***If you have any comments on this article, or suggestions for future material, please e-mail me at adri@n10net.com.***

§ ***I offer consultancy, technical support and mentoring services and DataEase-related products, using DataEase for Windows and ASP.NET with SQL Server. I am based in London, UK.***

About The Author: Adrian Jones is the former editor of 'Dialogue, the DataEase magazine', the creator of the 'template' applications that accompany DataEase 6, and a freelance DataEase consultant. A leading thinker in the use of the Windows product, he started with DataEase DOS 2.53 to solve the needs of his publishing business in the 1980s. His consultancy skills have been used by manufacturing, banking and charity organizations, and most recently by the Office Of Chief Medical Examiner in New York as part of the post 9/11 recovery. In July 2004, he received an MBE for 'services to British and local families in New York'.