Registry riddle

Mark Whitehorn asks: why isn't the Windows registry a proper database?

Question: *when isn't a database?* Answer: never. Yes, I realise that this sounds like a question from *Alice in Wonderland* but it has some degree of relevance when applied to Microsoft... *Oh no, he's going to have another rant at the big M!* — correct.

Most people who use one of Microsoft's operating systems will have noticed the registry. It is designed to hold information about the hardware upon which the OS is running and the software that's installed to run on it.

My argument is simple: the registry holds important data so it should be a proper database. If this were so, the registry would have features such as transaction control, roll back, decent backup and all of those good things... *Ah ha, he just wants those things because he is a database freak.* No he doesn't, he wants them because the registry screws up so royally on occasions because it does not have them. Consider the following and see if they sound familiar:

- **You install a program** on a Windows NT machine, the install program crashes, the registry is left in an indeterminate state and the NT box is unstable thereafter. Ultimately you have to re-install NT and all the other programs on the machine.
- **Your NT box** smokes itself — nothing to do with NT, it was a hardware fault — so it dies. Ah, but you have a backup! You find a spare box, restore from the tape and discover that NT won't run. Why not? Because the registry stores detailed information about the hardware and the spare box is not identical to the original.

Now, if the registry were to be managed as a respectable database, it could use roll-back to recover from a fluffed installation. And, if it were a sensibly-designed database, we could query it and separate the hardware and software data so that restores to different hardware became possible. But it isn't, so we can't.

You will gather from all this that I have had a traumatic week. To add piquancy, it was SQL Server 7.0 that screwed up the registry for me. So, '*When isn't a database?*' Never. Databases are designed to manage important data, so if you have important data, use a database.

There is no excuse for Microsoft but this also means that neither is there an excuse for the rest of us. If you are building an application which handles anything other than trivial data, think seriously about back-ending it with a database. You can embed the database engine so the users don't even have to know that it is there. They won't notice when it doesn't crash. In fact, you won't get much credit at all if it works but, hey, that is the lot of a good application developer.

If you think that I am the only person who ever suffered in this way with the registry, take a look at support.microsoft.com/support/downloads/dp3049.asp. There, you will find a tool known as RegClean.exe — I applaud the fact that Microsoft makes this tool available, but I am horrified that it needs to. I wonder whether NT2000 will be any better?

You will gather from all this that I have had a traumatic week

Item	TotalNoSold	TotalNoOrdered	StockLevel
Desk	3	220	217
Lamp	3	50	47
Chair	7	0	-7
Table	3	2	-1
Chest	60	0	-60
Bookcase	0	500	500

ItemNo	Item	Total_Sold	Total_Ordered	Stock_Level
1	Desk	3	220	217
2	Lamp	3	50	47
3	Chair	7	0	-7
4	Table	3	2	-1
5	Chest	60	0	-60
6	Bookcase	0	500	500

▲ **Fig 1** Two different ways of achieving the same result. One is elegant, the other more generally applicable; the choice is yours

When is a null not a null?

In our April column, I provided a solution to a problem sent in by Jason Holt <Jason@creasefield.demon.co.uk> which related to stock control. One of the problems which rears its head when dealing with this, and in all sorts of other situations, too, is how to handle null values.

I won't bore you by repeating the entire story but essentially there is a problem, say, when you subtract a null from a known value. The solution I provided used multiple queries to change nulls to zero.

Several readers have suggested more elegant solutions, notably: Ken Sheridan <KenSheridan@compuserve.com>, Alastair Bishop <ali@circle-k.demon.co.uk>, Peter Murray <pm4380@GlaxoWellcome.co.uk>, 'Fionnuala' <fb@nursingboard.ie> and Steve Devaney at steve@sdevan.demon.co.uk. All their emails and suggestions are on our cover disc in a text file called NULLS.TXT. These are well worth reading because most of these people have solved the problem in subtly different ways, all of which might be applicable for a situation you meet in the future.

[FIG 2]

A 'workdays' function

```
Public Function WorkDaysDiff(dtmLastDate As Date,↙
 dtmFirstDate As Date) As Long

  Dim lngWorkDaysDiff As Long

  If dtmFirstDate > dtmLastDate Then
     MsgBox "First date cannot be later than second↙
 date.", vbExclamation, "Error"
     Exit Function
  End If

  Do Until dtmLastDate = dtmFirstDate
     dtmLastDate = dtmLastDate - 1
     If WeekDay(dtmLastDate, vbMonday) < 6 Then↙
  lngWorkDaysDiff = lngWorkDaysDiff + 1
     End If
  Loop

  WorkDaysDiff = lngWorkDaysDiff

End Function
```

(Key: ↙ code string continues)

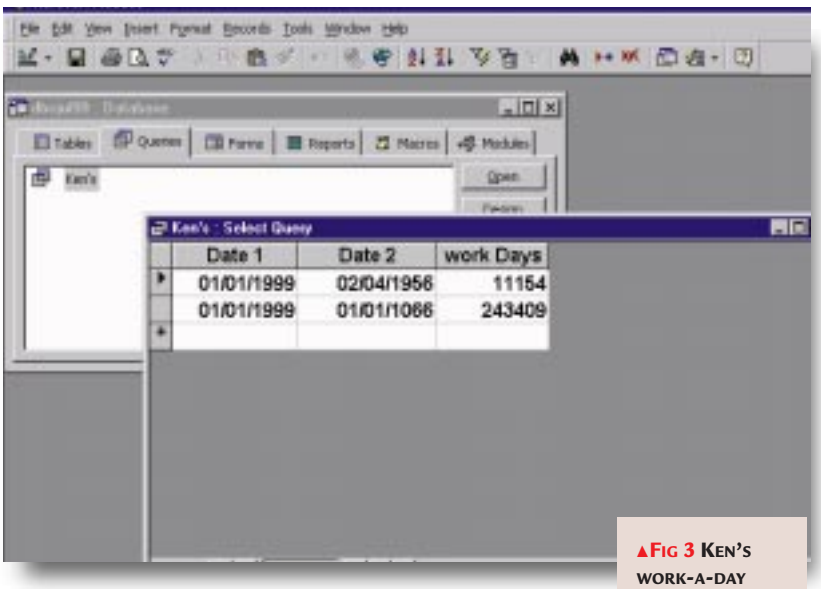
Why did I solve the problem in my more kludgy way? Well, the answer is that this is a databases column, not an Access column and I do try to supply answers for all readers [Fig 1].

Each of the supplied solutions make use of functions which are not, as far as I am aware, found in all RDBMS products. Some of them use the NZ() function which specifically returns a zero, a zero-length string (" ") or another specified value when fed with a null. Others use IIF() and ISNULL() to solve the problem, as in; IFF this value ISNULL then make it a zero.

My understanding is that NZ(), IFF() and ISNULL() are Access-specific — although the last two are reasonably common in other RDBMSes. The construction I used, namely 'Is Null', is an operator rather than a function which is, as far as I am aware, actually part of the SQL standard — it is certainly ubiquitous in RDBMSes.

In retrospect, I should have provided an Access solution as well — I do know about the ISNULL() function, honest, it's mentioned in the 'Inside Relational Databases' book — but I never expected so many readers to be so eagle-eyed!

Several readers supplied sample files. One, from Steve Devaney, is on our cover disc as DBCJU99.MDB. It also contains my original for comparison.



▲FIG 3 KEN'S WORK-A-DAY ALGORITHM IN ACTION

■ Working week

Reader, Ken Sheridan (*see above*) also supplied the following function in his email: his text makes it self-explanatory; 'Changing tack, you might be interested in this little function [the code is shown in Fig 2] to return the number of working days (Mon-Fri) when one date is subtracted from another. I've declared it as Long just in case anyone should want to know the number of working days since the Norman conquest.'

Ken's solution is conceptually simple and I wholeheartedly approve of simple algorithms because they are easier to handle and debug [Fig 3]. I have put it into an MDB called DBCJUL99.MDB on our cover disc. The only problem is that this one can be slow in practice because the algorithm requires each date in between the pair to be tested. If there are lots of data pairs to process, which happens if you use the function in a query, then the response can be tardy.

The registry holds important data, so it should be a proper database

However, when I read this email I realised that the problem sounded hauntingly familiar and knew that I had been here before. There are algorithms which are, admittedly, more complex than this but they are considerably faster.

Next month I'll publish the ones I know about, but meanwhile you may feel like trying to work out a faster answer.

PCW CONTACTS

Mark Whitehorn welcomes your feedback on the Databases column. Contact him via the PCW editorial office (address, p10) or email database@pcw.co.uk