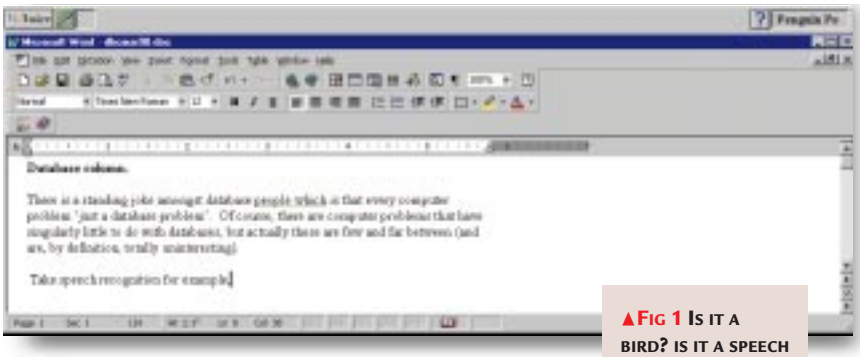
Database, where are you?

You'll find databases in the most unlikely places. Mark Whitehorn gives a few pointers.

There is a standing joke among database people, which is that every computer problem is "just a database problem". Of course, there are computer problems which have little to do with databases, but these are few and far between (and are by definition totally uninteresting). Take speech recognition, for example. As I speak these words, they are appearing on the screen courtesy of a database. True, it is a specialised one — IBM's ViaVoice, to be exact [Fig 1].



▲Fig 1 Is it a bird? Is it a speech recognition system? No, it's a database

Speech recognition matches sounds against character strings, all of which have to be stored, so it is a database problem. In fact, speech recognition is a relatively complex problem so ViaVoice doesn't only match sounds to text strings, it also looks at the strings in relation to each other. In other words, it looks at the context of the words and uses that information to alter the match of the string to the sound: it decides whether you are really saying "to","two" or "too" from the words that surround it.

Why is this relevant? Only that I recently spent time cleaning up some data (originally from a set of Word files) so that it could be imported into a database. The work was necessary because the data in the Word file was horribly haphazard; if a particular piece of data was missing, it was simply omitted rather than indicated by some form of null. Once the job was finished, I realised that most of my time had been spent ensuring that the correct data was ending up in the correct position in a Comma Separated File so that when it was imported, it ended up in the correct field. Now, suppose that I had access to a set of algorithms that could look at data in context: much of this work could have been automated.

Records inherently contain important positional data

It also made me think again about positional data. There is a science fiction story in which a character is left in charge of an electronic library. He promises to ensure the safety of all the words, and he does. The twist is that he sorts the entire contents of the library into alphabetical order, making the point that positional information is often more important than it first appears. It is worth remembering, as your database is filled, that records inherently contain positional data which may be important but may not be automatically recorded. Are you recording the fact that the data for a given record arrived before that in another? Does it matter that there was a six-second gap between the first five records and a two-hour gap before the next? I'm certainly not suggesting that this is always important, only that it might be, and that this sort of information is easy to record at the time but impossible to recover at a later date.

▼Fig 2 The two date data files ready and waiting to be converted — and ViaVoice is lurking at the top of the screen...

HERE IS THE NEWS

"Just to let you know about a new database-related newsgroup: comp.infosystems.www.databases. It has been created for the discussion of web databases and techniques. I thought your readers might like to know. I've had a look and it seems like it could be fascinating — it's well worth a look."

Guy van den Berg
guy@hole-in-the.net

Thanks for the information, Guy.

Hot dates

I am constantly touched by the forbearance and kindness of *PCW* readers. Take this letter, from Neville Kuyt <Neville.Kuyt@pa-consulting.com>:

"I am not too hot on Access, but I cannot resist. Your article on converting dates is naughty! Access is a relational database and databases are set-based. Using a loop to visit every record in turn is how we do things in VB: on a database, SQL should prevail. Apologies if this doesn't work in Access, but surely

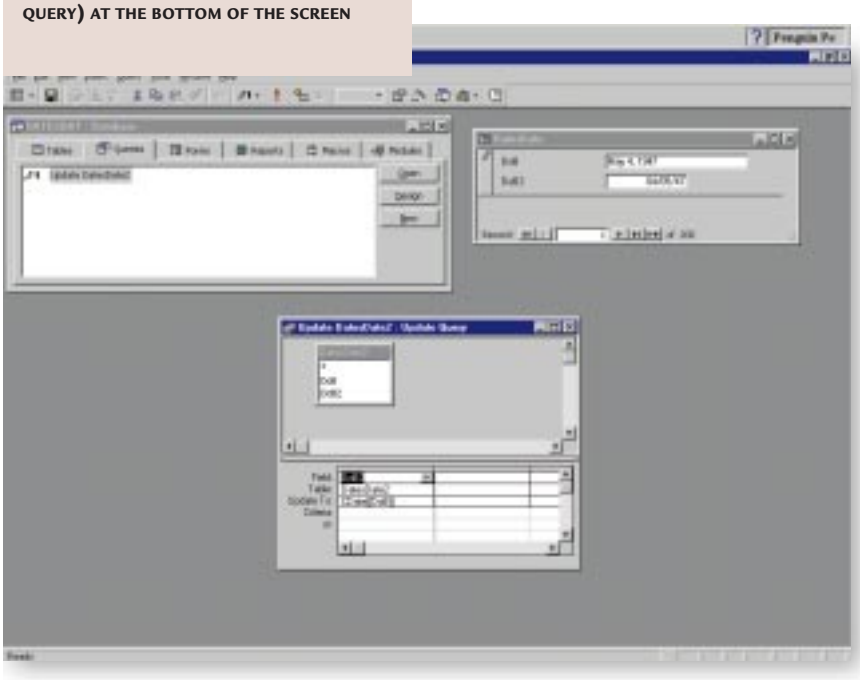
```
update DatesDate set DoB2 =
cdate(DOB)
```

is a far more elegant way to do this? Not only elegant, but quick, and "morally" correct. There are a lot of people out there who read your column, for whom the biggest benefit is to realise that SQL is about sets of data, not about loops. Just a thought... Apologies again if it doesn't work (I work with SQL Server where this works extremely well)."

Neville berates with such kindness, it sounds like a compliment. He is also correct. However, I have published many set-based solutions to similar problems in the past. As I said at the time, this was a fun solution rather than an optimal one. I quote from the December column: "Having been depressed by the idea that you might have to convert all the little charmers by hand, why not set up a form so that you can actually see them all being converted in front of your very eyes?" So, if you need efficiency — say, when you have 20,000 records rather than 200 — please do use a set-based solution.

In Access, a set-based solution is typically implemented as an update query. I have included the original solution and an update one, on the *PCW* cover disc. Copy this file to a hard disk but remember that it will copy across as "Read-only", so in Explorer use right-click, Properties to remove this restriction. Then have a look at the two tables, DatesDate and DatesDate2 [Figs 2 & 3]: note that the DoB2 field is empty in both. As before, the fun solution works via the form and operates on the table DatesDate. Open the form, put your cursor on the DoB2 field and scroll through the records. The Dob2 field updates to the correct value. Then run the query called "Update DatesDate2". This should update all 200 records in DatesDate2 at a stroke.

Reader, Nick Dowling <ndowling@lombard.co.uk> noted: "Your December column mentions that Access 2 does not have the CDate function. However, the DateValue function or the CVDate function will do what you need."



▼ FIG 3 ...AND TWO WAYS OF SOLVING THE PROBLEM. THE FORM/FUN WAY (TOP RIGHT) AND A SET OPERATION (USING AN UPDATE QUERY) AT THE BOTTOM OF THE SCREEN

AUDIT TABLES

Keeping corruption out of multi-user Access databases.

On the subject of corrupted access databases (November '98 column), reader Rufus Chapman <rufusc@bigfoot.com> writes: "I have also found Access to be fairly reliable despite recent disclosures of things like the Access bookmark sync bug (which, incidentally, would affect Chris Veness' combo box example in tables with more than ~262 records. Perhaps you should note this in your next column?). But in some environments, notably where multi-user access is combined with users who see it as their job to try and break your databases, I find that some record and/or file corruptions do occur. In my experience, file corruptions are rare, but in some cases I've found that either some combination of users pasting strange things into fields and/or poor programming/testing on my part (guilty!) leads to some record corruptions where a particular field value shows up as #ERROR and you can neither Compact nor Repair the database. The only way round this is to delete the record and then backup/repair and reinstate it. The best way I've found of doing this, basically imitating a checkpoint of a larger-scale database like Ingres or Oracle, is to write an 'audit' table." Rufus goes on to explain how the code works and generously includes his code. This is too long to include here, but his email, including the code, is on our cover disc as a text file.

PCW CONTACTS

Mark Whitehorn can be contacted via the PCW editorial office (address, p10) or email database@pcw.co.uk