Tricks and **treats**

Mark Whitehorn lets you in on some secrets for writing Visual Basic code in Access and unravels the mysteries of string manipulation. He tackles your knotty problems as well.

In my November column, reader James Talbut proposed a way of formatting dates using ordinal numbers ("Monday 2nd June 1997") but wondered if there was a faster or more elegant solution. Anyone care to speed-test these two suggestions and report back?

1. Wilf Davies sends in one possible solution, shown in Fig 1.
2. The other, shown in Fig 2, from Ray Hall is certainly more elegant but I don't know if it's quicker. There are fewer comparisons but I don't know how much overhead is carried by the Case statement. Declarations as printed, except sRetVal.

Writing code in Access

This month, by popular request, we're going to have a look at some tips and tricks for writing Visual Basic code in Access. This is going to be based on Access 97. I am aware that lots of readers are still using Access 2.0, but the coding languages of the two versions are somewhat different and I had to choose one or the other, so I went for the later version.

How to start programming in Access

There are many ways to get started but this is probably the easiest. Start in a database that has a few tables of data (if the data has any value to you, make sure you are working with a copy. There is a sample database on our cover-mounted CD-ROM this month which provides some samples that you can use in safety).

Create a blank form and open the toolbox. Make sure that the control Wizards are turned off and then place a button on the form (Fig 3). Right-click on the button, select Build Event, choose Code Builder and you are ready to code (Fig 4). Whatever code you write will be carried out (executed) whenever the button is clicked.

String manipulation

Strings are long sets of characters. For example, the characters in this sentence make up a string. Indeed, all of the characters in this paragraph make up a larger string. Programming languages in general, and Visual Basic in particular, have functions which are designed to manipulate strings. We'll have a look at three.

1. How to find the length of a string

Len(string) will tell you the number of characters in a string. For example:

```
Length = Len("Penguin
and his friend.")
```

will set the value in the variable Length to be equal to 23. Of course, there is nothing to stop you setting the value of, say, a text box on a form to be equal to the length of a

Fig 1: Wilf Davies' solution

```
Public Function OrdinalDate(DateIn) As String
    Dim DayNo As String, Ordinal As String
    If IsNull(DateIn) Then Exit Function
    DayNo = Right$(Str$(Day(DateIn)), 1)
    Select Case DayNo
        Case "1"
            Ordinal = "st"
        Case "2"
            Ordinal = "nd"
        Case "3"
            Ordinal = "rd"
        Case Else
            Ordinal = "th"
    End Select
    OrdinalDate = Format$(DateIn, "dddd d") & Ordinal & " " &
Format$(DateIn, "mmmm yyyy")
End Function
```

Fig 2: Ray Hall's solution

```
Dim Suffix As String
Select Case iDay
     Case 1 To 3     Suffix = vOrdinals(iDay)
     Case 21 To 23   Suffix = vOrdinals(iDay Mod 10)
     Case Else       Suffix = "th"
End Select
FormatDate = Format(dArg,"dddd, d") & Suffix & Format(dArg, "mmmm,yyyy")
```

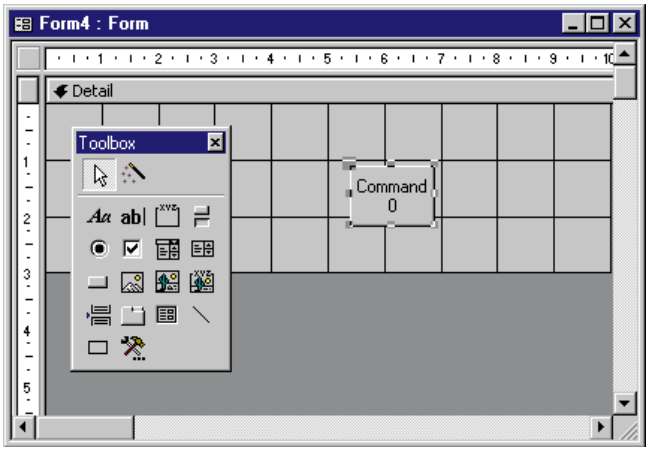
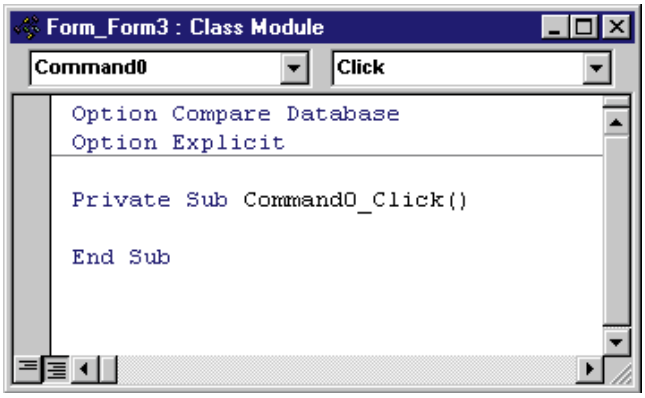


Fig 3 (left) Putting a new button on to a form

Fig 4 (below) Code written in this window will be executed whenever the button is clicked



string held in a field. Assume that we have a table in the database called Sample and that we have set up rstSample to reference that table. Sample contains a field called Information that in turn holds a string "Penguin and his friend." In that case, the code

```
Text3 = Len(rstSample![Information])
```

will set the value of a text box called Text3 to be 23.

● There are many other tips in this area so I have included them on this month's CD-ROM: see the form called "Remove string from table and manipulate it".

■ How to find the position of a character in a string

InStr is a function that will find the location of a particular character in a string. Thus, Fig 5 will set the value of the text box called Text7 to a value of 2. The 1 tells InStr to start searching at the beginning of the string. Thus, Fig 6 will set Text9 to be 20 because it starts the search at the 3rd character into the string.

■ How to extract a copy of part of a string

Mid is the function we need here. You need to give Mid the string with which you want it to work, the number of the character at which it should start copying, and the number of characters it should remove after that start character. So Fig 7 removes seven characters from the string, starting at the third. So it will set Text5 to be equal to "nguin a".

Even though all these functions may seem a little abstract at first, the reason I have chosen them is that they all help to answer particular queries I have received from readers.

Fig 7

```
Text5 = Mid(rstSample![Information], 3, 7)
```

Fig 5

```
Text7 = InStr(1, rstSample![Information], "e", 1)
```

Fig 6

```
Text9 = InStr(3, rstSample![Information], "e", 1)
```

A couple of more general tips

1. Modifying your start up

Access typically starts up with the words "Microsoft Access" in the top left of the screen (in the title bar). You can configure this to read anything you want: boringly, the name of your application; more excitingly, a short description of your boss (don't blame me if you get fired). Simply right-click the title bar of the database container (not the title bar of Access itself) and select Startup. In the dialog

p292 ➢

Questions & Answers

Q My friend has a DOS database which stores its data in three text files. I've managed to import these into Access 97 (though I did get some field truncations) and now have three tables arranged [as shown in Fig 8]. In practice, the Cross Ref field can contain many more values than are shown here and Access truncated some of the data. Advice on how I can reduce the truncation of the text field would be appreciated: Access would not import this as numeric, presumably because of the commas between items. Also, my friend wants to look up anyone listed under, say, Anteater Supplies and extract the data for Anteater Bedding Inc., Ants R Us, and Everything Anteater Inc., (i.e. items 2, 3 and 4093). Is there a way to link these tables?

Nigel Mercier

A The truncation is easy, but make sure you import the data into a memo field. Memo fields can hold a minimum of 65,656 characters, so that should be enough.

The problem with the data in the "Cross Ref" field is that it is not atomic. That is, a single field in a single record is actually holding multiple individual bits of information. We need to turn the data into the sort of table shown in Fig 9. This can be done by opening up the table called Cross Ref, dissecting the string in the Cross Ref field and building new records in a new table called NewCrossRef. This is a good example of where the ability to manipulate tables with code is really useful. I can't think of a way to do this manipulation using, say, SQL.

A solution using code is available attached to the form called "Solution to problem". Note that this code actually uses a table called CrossRef (no space). This is just like Cross Ref (with a space) but contains a long string so you can see it working on more realistic data. There are three sample text files included on our cover-mounted CD-ROM so that you can play around with importing if you so desire.

Q What code do I need to manipulate databases?

A Access code uses variables to hold the names of the databases that you are going to manipulate with the code. You declare variables at the start of the code by using the word DIM (DIMension) which implies that you are "dimensioning" or setting up a space to hold information. (If this all sounds like Venusian, don't worry. When you are starting out you can follow the examples and do it by rote.)

So, to gain code-type access to a database you would use:

```
Dim DBCurrent As Database
Set DBCurrent = CurrentDB()
```

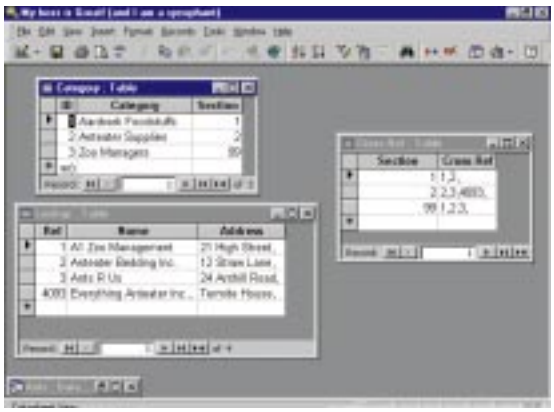
The first line says "set up a variable called DBCurrent and make that variable capable of referring to any database". The next line says "tie that variable name to a particular database, in this case the one which is currently open". Any reference in the code to DBCurrent now automatically refers to the current database.

You can substitute almost any word for DBCurrent since it is a variable name. However, the rest of the words are fixed. This code is demonstrated in the sample database on the form called "Open a database".

Q How do I reference a particular table in a database?

A First of all, you have to set up the database (*as shown, above*). Once that has been done, you can use a variable to refer to a table.

The variable in this case is often referred to as a record set. Thus:



NewCrossRef ...

	Section	Cross Ref
▶	1	1
	1	2
	2	2
	2	3
	2	4093
	99	1
	99	2
	99	3

Record:

Fig 8 (top) and Fig 9 (above) The crossRef field needs to be in atomic format

```
Dim rstCrossRef As Recordset
Set rstCrossRef = DBCurrent.Open
Recordset("CrossRef", dbOpenTable)
```

is a long-winded way of saying that from now on, in this block of code, any reference to rstCrossRef is actually a reference to a table called CrossRef in the current database.

This does, of course, assume that the current database contains a table called CrossRef. See the form called "Open a database and a table" on this month's cover-mounted CD-ROM.

that appears you can type the heading of your choice into the Application Title box.

1. Loadsa free toys

For those with access to the web, there is a raft of fun free stuff available for Access at www.microsoft.com/officefreestuff/access/. This includes the long overdue Wizard that lets you print out your relationships without using multiple screendumps and other messy workarounds.

Those with experience of www.microsoft.com will know that you are well advised to download material while America is in the land of nod (approximately 8am to 1pm, our time).

PCW Contact

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column. Write to him at the usual *PCW* address (see page 12) or email him at database@pcw.co.uk

Hot dates

Mark Whitehorn presents advice on Access dates and nested ifs. He promotes the advantages of a good view in client-server, with permissions, to get exactly what you want.

Reader, Simon Faulkner, has sent in a batch of date functions for Access. He writes: "Here are a few functions which I wish were built into MS Access. I used to spend hours trying to achieve them in queries etc., and I think these are the simplest ways. The FirstOfMonth and LastOfMonth are especially useful for invoicing." (Fig 1)

Making the grade

In my September column, I provided a set of nested "if" statements which provided grades for students' marks. Reader, Peter Vize, says he has a better solution: "I don't like nested ifs in criteria fields in queries. It provides problems with documenting, error checking and maintenance (for example, if the grade structure changes).

"My solution, which I attach (in Access 7.0 unfortunately — thank you, Office 97!) is to create a grade lookup table and generate a query with the student table without any links between the tables (a Cartesian join)."

The file is on this month's CD-ROM as PETER.MDB. I like Peter's solution; in fact, I would say it is better than mine. You might want to check it out. For those without Access, or with version 2.0, the SQL reads as shown in Fig 2 (*p280*). Peter's solution uses the same tables as the solution published in the September issue, except that he has added the table shown in Screenshot 4 (*p280*).

Client-server views

Continuing the topic of client-server systems from last month, we come to the topic of Views. In a database, data is stored in tables. Permissions, which allow or deny access, can be given to users and groups of users for each table. The permissions can be varied for a given table: one user might be allowed to see the data but not edit it, while another could edit but not see the data. A better option would be to let another user see the data *and* edit it.

However, most RDBMSs do not allow variation of the permissions within a table, and therefore a user can either see everything in a table, or nothing, and there are many occasions when this is not ideal: views provide a flexible way of letting users work with the data in a database.

(*p280* ➢)

Fig 1 Date functions in Access

```
FirstOfMonth = Now()-Day(Now())+1
LastOfMonth = DateAdd("m",1,Now()-Day(Now()))
FirstOfYear = Now()-DatePart("y",Now())+1
LastOfYear = DateAdd("yyyy",1,Now()-DatePart("y",Now()))
DaysInMonth = DatePart("d",DateAdd("m",1,Now()-Day(Now())))
DaysLeftInYear = DateAdd("yyyy",1,Now()-DatePart("y",Now()))-Now()
DaysLeftTillChristmas = DateAdd("yyyy",1,Now()-DatePart("y",Now()))-Now()-7
DaysInYear =
DateAdd("yyyy",1,Now()-DatePart("y",Now()))-(Now()-DatePart("y",Now()))
LeapYear =
IIf((DateAdd("yyyy",1,Now()-DatePart("y",Now()))-(Now()-DatePart("y",Now()))
))=366,True,False)
```


Questions & Answers

Q I come from the old-fashioned world of DOS-based machines [*I remember my grandfather talking about those — MW*] and yearn for reverse video on my forms. Can it be set during form design as a property in a text box?

Sarah James

A Not directly, but you can get the same effect by selecting the field (or fields), calling up the palette, and setting the fore colour to white and the back colour to black.

Q Okay, I'll admit to being confused about Yes/No formats. I seem to be able to enter "Yes" or "No" and the data is stored as minus one and zero respectively — this I can cope with. It won't accept input such as male/female or big/small, but, I recently discovered, it will accept on/off. What are the rules which control this and what is the best way to use this field type?

Brian Clarkson

A Access supports a range of data types, one of which is the Yes/No type. What makes this particular data type potentially confusing is that you have to keep in mind three separate components: the input, the storage and the display format.

1. Input

Yes/no data types will accept the following input:

```
Yes, No
On, Off
True, False
-1, 0
```

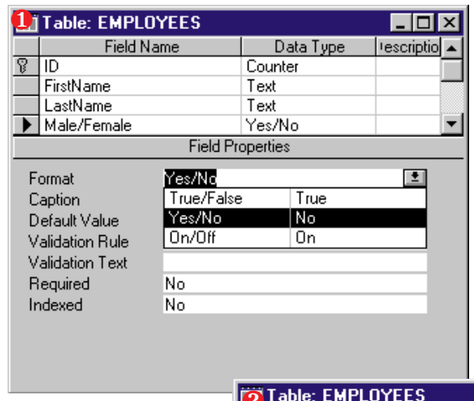
(or input via an interface control such as a check box).

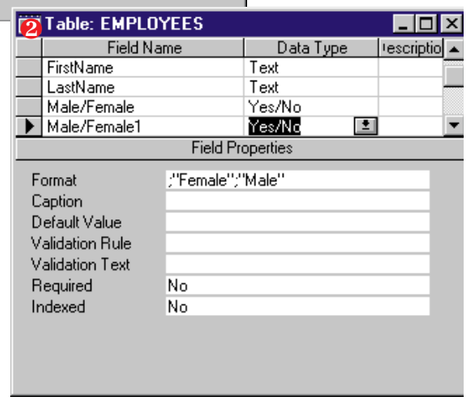
2. Storage

Whatever the input, the data is stored as -1 or 0. (Do any readers know which bit of computer history lead us to why -1 is regarded as true, while zero is false?)

3. Display

By default, whatever input is used, Access displays -1 as Yes and 0 as No. However, you can select an alternative format in table design (Screenshot 1). You can also produce a customised output that will display virtually anything you like for the values –1 and 0. For example, to display the





Screenshots 1 & 2
Setting up your field formats to produce customised outputs couldn't be easier

p278 ➢

male/female that Brian suggests, you simply need to write:

```
;"Female";"Male"
```

(*note the leading semicolon which is easy to miss!*) as a custom format (Screenshot 2, p277). This tells Access to display the word "Female" where the field contains –1, and "Male" where it contains 0. You can add colours, as well:

```
;"Female"[Blue];"Male"[Red]
```

However, the bad news is that this doesn't mean you can type the word "Male" into the field and expect Access to store 0, because all we have altered is the display format. If you try to enter the word "Male", Access sees it as a text string and refuses to store it in the Yes/No field.

Brian's next question (hard on the heels of my email reply) was:

Q Okay, that makes sense. But in that case, how *do* I get Access to let me type in "Male" or "Female" while ensuring that no other information gets in?

A I wouldn't use a Yes/No field. First, I'd set up a text field and set the Validation Rule property to be:

```
In("Female","Male")
```

Two more useful (but not essential) options that you can set are Default and Required. Set the Default value to whichever value is more appropriate, and the Required property to "Yes" if you must have a value in the field.

In Access 97 you can do all that, but you can achieve the same end result by using the lookup tab as shown in Screenshot 3. The advantage of this method is that you will be automatically creating a combo box for your forms and datasheets, which makes data entry even easier for the user.

Screenshot 3a shows a sample table (in Access 2.0 format) which is included on this month's cover CD-ROM and provides samples of most of the aforementioned options.

Q I want to develop a database that uses Selective Dissemination of Information (SDI). I have a few thousand company names and addresses that I want to match against incoming data. The nature of the profiles (text strings) means the match must be relatively fuzzy. There are some products on the market (ZyFilter, Verity Agent Server) but they are a little out of my league. Is there a Windows desktop database that could do this kind of thing?

Peter Sells

A Products like AskSam allow you to search through unstructured data <www.asksam.com>. I've used this product in the past and it may be close enough to what you need; it is at least worth looking at.

As a more general point, you can use wildcards in Access to search for "fuzzy" data. The following variations are useful in queries (none are case sensitive):

- **Like "Ba?"** — Finds all records where the field in question contains "Bar", "Baa", "Bal" etc., but not "Baaa".
- **Like "P*"** — All records where the field starts with "P", irrespective of text length.
- **Like "*Penguin*"** — All records where the field contains the word "Penguin" anywhere in the text.
- **Like "*/02/*"** — All records that appear in February of any year.
- **Like "[AEIOU]*"** — All records where the field contains text starting with a vowel.
- **Like "[!AEIOU]*"** — All records beginning with a consonant.

The last two variants don't have to be used in conjunction with vowels and consonants. [!ABCD]* means starting with any letter except ABC or D.

Q I want to find all the records which are between two values — say, people with between two and four children.

Paul Brown

A Surprisingly, you've almost answered the question yourself, since Access provides an operator called BETWEEN which does precisely this. You can use it in a query like this:

```
Between 2 And 4
```

Q In my work, we have a successful Access database that I wrote a while ago. We now want to make it so that several people can access (joke!) the data at the same time. We have a Novell 3.11 server. How do I do this?

3 EMPLOYEES : Table

Field Name	Data Type	escriptic
Male/Female1	Yes/No	
Male/Female2	Yes/No	
Male/Female3	Text	
Male/Female4	Text	
Date	Date/Time	

Field Properties

General | Lookup

Display Control	Combo Box
Row Source Type	Value List
Row Source	Male;Female
Bound Column	1
Column Count	1
Column Heads	No
Column Widths	
List Rows	8
List Width	Auto
Limit To List	Yes

3a

	FirstName	LastName	Female	Male/Female1	Male/Female2	Male/Female3	Male/Female4	Date
1	Fred	Marsh	No	Male	Male	Male	Male	01 February 1945
2	Betty	Graves	Yes	Female	Female	Female	Female	05 June 1978
4	Penguin	Penguinsson	No	Male	Male	Male	Male	21 March 1992
5	Harry	Logan	No	Male	Male	male	Male	03 February 1989
6	Phil	Greeves	No	Male	Male	Male	Male	
7	Sarah	Lomax	Yes	Female	Female	Female	Female	
8	Shelia	Rogers	Yes	Female	Female	Female	Female	
9	Bill	Harrod	No	Male	Male	Male	Male	04 June 1978
*	ber)		No	Male	Male	Female		

p280 ➢

A The simple answer is to move the .MDB file on to the file server. As multiple copies of Access (each loaded on a workstation) access the data therein, the .LDB file will look after the potential locking problems. But (and it is a *big* but) you will find that this will place a heavy load on your network if the database is complex or has too many users.

Who cares about network loading? Well, everyone; or at least, all the users who will have to cope with a drastically slow response time. Probably the best solution, technically, is to move to a client-sever system but that is expensive all round, in terms of hardware, software and development work.

A useful work-around is to separate the data and the interface components into two .MDB files. The .MDB file containing the data can be mounted on the file server (where its .LDB file will still manage the locking issues). The .MDB file containing the interface components, macros, queries and forms can be placed locally on each workstation.

This has several advantages. The first is that only the data (not the forms, queries etc.) needs to be moved across the network, so the traffic is significantly reduced. Second, by separating the data from the interface, you have made it much easier to update the interface section. You can also give different users different interfaces more easily. Finally, backup should be faster, because only the .MDB file on the file server will need to be backed up and that will be smaller than a combined file.

The only disadvantage is that you need to perform separation of the data and interface bits. Access 97 provides a Wizard to do the job for you. In Access 2.0 I would do it by making two copies of the original file. I would delete the interface components from one, the data tables from the other, and then make the attachments between the two .MDB files (using File, Attach and then browsing to the appropriate Access file and tables).

Remember to take several backups of the original .MDB file *before* you start playing around with it. ■

Fig 2 SQL for Peter Vize's grade lookup table (*see p276*)

```
SELECT DISTINCTROW STUDENTS.FIRSTNAME, STUDENTS.LASTNAME, COURSES.COURSENAME, ATTEND.Exam, ➢
[Grade Table].GRADE
FROM [Grade Table], COURSES INNER JOIN (STUDENTS INNER JOIN ATTEND ON STUDENTS.ID = ATTEND.ID) ON ➢
COURSES.CODE = ATTEND.Code
WHERE (((ATTEND.Exam) Is Not Null) AND (([Grade Table].MAX)>=[Exam]) AND (([Grade Table].MIN)<[Exam]))
ORDER BY [Grade Table].GRADE;
```

(Key: ➢ Listing continued on next line)

4 Grade Table : T...

MAX	MIN	GRADE
999	70	A
70	65	B
65	60	C
60	50	D
50	40	E
40	0	F
0	0	

Record: 1

Suppose you have a table of staff details. Clerical staff need to see data about the personnel employed (their department, the date they started work, etc) but as the table also holds salary, performance and health records, unrestricted access cannot be allowed. A view lets you extract a subset of the fields from a table. Access to views can be controlled with Permissions in the same way that access to tables is controlled. It is also possible to create multiple views on the same table: one which allows personnel staff access to everything except medical details; and another for the medical officer, which allows access to medical records only. In this way, views can be used to provide the fine control needed to maintain security. In many databases it is common to use controlled access to views as the sole way for users to access data.

What's the difference?

Views can not only subset data by field, but also by row, and are typically defined using SQL. You may wonder whether there is a difference between a query and a view? If you come from an Access background this is a fair question, because Access doesn't distinguish between views and queries: it provides only queries but allows you to use them as if they were views.

So what are the differences between views and queries?

- In general, DBAs (Database Administrators) create views and users create queries (there are exceptions to this, but it stands as a general distinction).
- Views are stored in the database itself (in fact, in the system catalog tables). Queries are stored in the front-end application.
- The result set generated by a query is not updatable. The data displayed in a view is often updatable.

The crucial difference is that in a client-server system, there is a greater distinction between the work of the DBA and the end-user. If you think of views simply as queries like those listed below, you have a reasonable working definition of a view:

- definable only by the DBA;
- safe from user interference;
- storable in the data dictionary; and
- capable of producing editable record sets.

Views have the added advantage of being able to hide the structure of the database from users, giving them a simpler interface with the data. Views can be created which pull together data from different tables, presenting users with what looks like a single table that contains everything they need for the task in hand.

Views are wonderful. You can use them, along with permissions, to provide exactly the correct information to precisely the right users; to de-normalise the version of the data that is presented to the user, making it clearer and more readable.

PCW Contact

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column. Write to him at the usual *PCW* address (on page 12) or email him at database@pcw.co.uk

Trigger happy

Mark Whitehorn fires off a few shots of wisdom about the new concepts in client-server systems, and updates you on the Bloor research report on Scalability in Databases.

"The wonderful thing about Triggers, is Triggers are wonderful things" (with apologies to AA Milne.)

Client-server systems are becoming easier to install and use. They bring with them a whole raft of new ideas, concepts and jargon that *must* be absorbed by an aspiring database professional, and we will look at some of these newcomers over the next few months.

The information presented here is not meant to be product specific, so it should apply generally to SQL Server, DB2, Informix, Sybase etc. Clearly there will be some implementational differences, but those should be relatively minor.

PC-based RDBMSs like Access combine the front-end and back-end into a more-or-less seamless whole. So when you want to ensure that a business rule is enforced within an Access database, it is common to put the code that enforces that rule into, say, a form. In a client-server system, the front-end application is divorced from the back-end database. This separation may be so complete that the person designing and building the back-end may have no idea what front-ends will be run against their work in the future, so it is necessary to have a mechanism for enforcing business rules within the database itself.

Incidentally, this certainly is not meant to imply that front-end applications on client-server systems cannot, or should not, enforce business rules. You might well want to apply different business rules for different groups of users, and one way to do this is to supply different front-end applications which apply different business rules. However, the big advantage of rules enforced on the server is that they cannot be subverted by any front-end application and so they are applied absolutely.

There are several ways of implementing business rules within a back-end database; triggers are used by many RDBMSs. Triggers are made up of parts, and the parts may differ slightly between implementations, but typically you may find that a trigger has: a name, a triggering operation (i.e. the event that sets it off), an activation time, granularity, a body, and conditions. These parts are all entirely logical as soon as you think about what a trigger is supposed to do.

The easiest way of explaining one is to look at an example. Suppose you have a (very sensible) business rule which states that the details of a given customer must not be deleted from the customer table if that customer still owes money to your company.

This trigger has to look for every SQL statement that tries to delete customers from the CUSTOMER table. The trigger must operate before the record is deleted. It must first check that the customer is debt free and only allow the deletion to occur if that is true. Finally, if the delete statement tries to delete multiple records at a time (DELETE * FROM CUSTOMERS) in a single statement, then the trigger must run multiple times, checking the debt status of each customer that the DELETE statement tries to remove.

The parts outlined above can be tied to a description of this trigger. It might be called Check_CUST_Del (name). It looks for any DELETE operation that occurs against the table (the triggering operation). When that occurs, the debt check is carried out (the body) before the DELETE completes (the activation time; the trigger is activated before the delete). If the DELETE runs against multiple rows in the table, the trigger is run once for each row in the table that's affected (the granularity).

If we modified this trigger so that it only applied the check if the customer had placed orders of a value greater than £10,000, then that would be a condition. Defining a condition for a trigger is optional.

More on Bloor

Those of you who have been worrying about the implications of the Bloor report on Scalability in Databases (see Hands On Databases, August) can stop worrying; the report essentially no longer exists.

So what about the facts described in the report? They do not appear to have changed. Bloor did find that Microsoft's SQL Server did not scale well, did find the other points of concern that were covered in my August column, and the company still stands by those findings.

What has changed is that the report itself is no longer an available part of the sum total of human knowledge. If you do not have a copy of it, your chances of obtaining all the information therein are slim indeed.

Here are some more facts, so far as I have been able to ascertain them. The report looked at RDBMSs from two companies: IBM and Microsoft. Bloor claims that both companies provided software and support people to set up the RDBMSs for testing; Microsoft disputes this, and claims that it did not supply people to help with the test.

The licence for SQL Server says (in broad legalese) that you must have Microsoft's written permission before publishing the results of testing. Bloor did not obtain this permission, and says that it took Microsoft's assistance in the project as tacit acceptance of the fact that the results would be published.

A couple of months after publication of the report, Bloor received a "cease and desist" letter from Microsoft's lawyers, which states that Bloor is in breach of contract because it failed to obtain the required written permission. Since Bloor had indeed failed to do so, it felt it had no option but to cease distribution of the report. A reasonable person might ask why, if Microsoft considered that the report was inaccurate, it didn't try to prevent the report being published on those grounds rather than on what appears to be a technicality?

The greatest loss from all of this may not be the information contained in the report, but the methods it used for testing databases. Testing for speed is a complex area, too often treated in a trivial way. This Bloor report was (in my opinion) a genuine contribution to our attempts to derive a

p288 ➢

Business rules

Databases have all sorts of generic integrity checks that can be applied: referential integrity, not nullable, and so on. These are "generic" integrity rules in that they have the potential to be applied to all databases. In addition to these generic integrity checks, there are constraints that you may wish to apply to a specific database.

It may be a rule in your company that no-one below the level of a manager can be given a pay rise of more than ten percent. It may be a rule that all orders with a value of more then £10,000 have to be approved by the MD. These are known as "business rules". They can be just as important as the generic varieties but clearly have to be handled by some other mechanism, such as triggers.

Tips and Tricks: Access shortcuts and ordinal numbers

■ Shane Devenshire is a regular contributor to Tips and Tricks. He lives in Walnut Creek, California.
In Access 95 or 97, here are a few shortcuts you might find interesting:

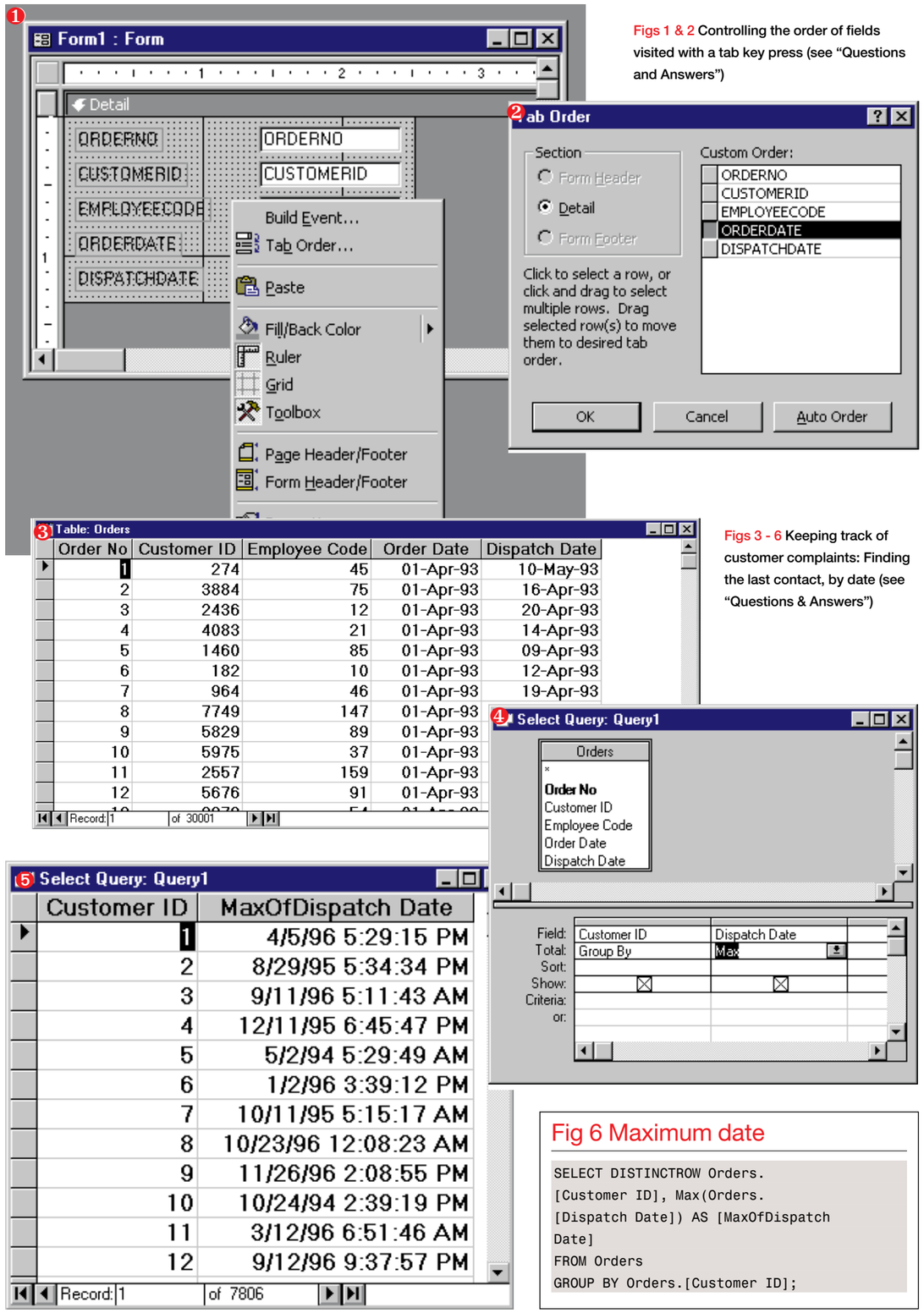
In the File Open, Save or Save As dialog boxes:
1. F4 drops down the Save In or Look In list.
2. F5 refreshes the screen.
3. Del (Delete) deletes a file to the Recycle bin.
4. Shift+Del deletes a file without sending it to the recycle bin.
5. Single-click the file, then single-click it again; this allows you to rename the file.
6. Right-clicking the file will display a shortcut menu with, among other things, Quick View and Send To.
7. With the Details option on, you can click the top of the columns to sort each field.
8. With Details on, you can also best fit the columns by double-clicking on the line between the column titles.

■ James Talbut came up with a problem and a solution, but wonders whether anyone knows of a better version?
"I would like dates to be formatted using ordinal numbers, hence 'Monday 2nd June 1997'. I cannot find any way of using number formats to produce ordinal numbers and have had to resort to the function shown (*see opposite*) which is neither quick nor elegant. Any comments?"

```
Public Function FormatDate(dArg As Date) As String
Dim sRetVal As String
Dim vOrdinals As Variant
Dim iDay As Integer
  vOrdinals = Array("th", "st", "nd", "rd")
  sRetVal = Format(dArg, "dddd, d")
  iDay = Day(dArg)

If iDay < 4 Then
    sRetVal = sRetVal + vOrdinals(iDay)
  ElseIf iDay < 20 Then
    sRetVal = sRetVal + "th"
  ElseIf iDay Mod 10 < 4 Then
    sRetVal = sRetVal + vOrdinals(iDay Mod 10)
  Else
    sRetVal = sRetVal + "th"
  End If

  FormatDate = sRetVal & Format(dArg, " mmmm, yyyy")
End Function
```



Figs 1 & 2 Controlling the order of fields visited with a tab key press (see "Questions and Answers")

Figs 3 - 6 Keeping track of customer complaints: Finding the last contact, by date (see "Questions & Answers")

Fig 6 Maximum date

```
SELECT DISTINCTROW Orders.
[Customer ID], Max(Orders.
[Dispatch Date]) AS [MaxOfDispatch
Date]
FROM Orders
GROUP BY Orders.[Customer ID];
```


Questions & Answers

Q How can I transfer stock between two locations in a firm chosen from a number of available options? All the names of the locations are stored in one table as different fields.

Kieran Gilmurray
(*who also supplied the answer*)

A This case statement (right) allows the user to do two things:
1. Determine the value in a field, in this case a quantity of stock residing at a particular location of their choosing.
2. Determine the actual name of the field itself to transfer the stock to. (A similar piece of code is used to transfer stock from a particular location.)

A case of transferring stock (*see question, left*):

```
'*** CASE TO BE DETERMINED WHERE THE STOCK IS TO BE ADDED TO ********
TOO = Me.[TOLOC] ' field on form containing location to which the
stock is going.
Select Case (TOO)  ' Test input.
Case "BURGESS HILL"    ' NB THIS IS THE SAME AS BURGESS HILL
TOOAMT = STOCK_RS![PHY_QUANTITY] 'amount in a field
TOFIELDNAME = "PHY_QUANTITY" ' name of the actual field in a stock
table
Case "PF_COURIERS"
TOOAMT = STOCK_RS![PF_COURIERS]
TOFIELDNAME = "PF_COURIERS"
```

And as many other "Case" lines as you need…

```
Case Else  ' Must be something else.
MsgBox "You have entered a location that is not present amongst your
available options", kstopsign, "Message"
Me.[TOLOC] = ""
Exit Sub
End Select
' ** use the field values and names in a calculation and update the
appropriate fields in your stock table
STOCK_RS.LOCKEDITS = False 'OPTIMISTIC
STOCK_RS.Edit          ' Enable editing.
STOCK_RS(TOFIELDNAME) = STOCK_RS(TOFIELDNAME) + Me![quantity]
STOCK_RS(FROMFIELDNAME) = STOCK_RS(FROMFIELDNAME) - Me![quantity]
STOCK_RS.Update        ' Save changes.
```

Q When I create a join in Access, that join appears in the query tools. Sometimes, even though I haven't created a join, one still appears in the query tool. Why?

A The rule is quite straightforward. A join will form automatically if:
- two tables appear in a query which have fields of the same name; or
- the fields are of the same type; or
- one of the fields is a primary key.

If you don't want the join, simply click on the line that indicates the join to highlight it and press the delete key.

Q What happens if there are no joins between a pair of tables in a query?

A The result is a Cartesian product. Suppose that you have six customers in one table and 40 orders in another. Assuming that each order has a reference to a customer, then you can expect a query that shows customer name and order information to have 40 rows. If you don't have a join showing between the two tables, you will get 240 rows (6 * 40) because each and every customer will be matched to each and every order.

As a general rule, Cartesian products are not what you want, so make the joins explicit by using the Relationship editor.

Q How can I control the order in which the fields are visited when the user presses the tab key?

A The default "tab order", as it is called, is the order in which the fields were added to the form. You open the form in design mode, right-click the form and select tab order (Fig 1, opposite). This opens up a box where you can visually set the tab order (Fig 2). This same box can be reached in Access 2.0 by opening the form in design mode and selecting Edit, Tab order from the menu system.

Q We are designing a simple Access database to keep track of customer complaints. The overall design was simple and it's about ready to use apart from a single report that I can't figure out. Each customer record has numerous contact records attached. How do I get a report that contains just the last (by date) contact data for each customer?

Ian Batterham

A The easiest way is to base the report on a query that finds the last contact data **should this be date?** for each customer. I happen to have an ORDERS table (which lists customer numbers and order dates) which serves to illustrate the method (Fig 3).

What you need is a "group by" query (Fig 4) which groups the customer numbers and finds the "maximum date": in other words, the most recent (Fig 5). The SQL for this is shown in Fig 6.

sensible methodology of testing databases.

I think that if Microsoft had known the report to be inaccurate, it should have attacked on that basis. But to attack the report on a technicality, rather than on technical grounds, seems to me the worst of all worlds, making the company look (in my personal opinion) ridiculous, high-handed and arrogant.

As I do not want to receive any "cease and desist" letters myself, I should stress that I am not suggesting Microsoft has behaved in an unreasonable way in this matter. I have every confidence that you can continue to think of Microsoft as one of the family: like, say, a brother… a big brother.

PCW Contact

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column. Write to him at the usual *PCW* address (on page 12) or email him at database@pcw.co.uk.

Return to client-server

Client-server upgrading, currently a hot topic among readers, takes centre stage. There are Access tips and tricks, plus... how can this be? Mark Whitehorn reviewing his own books?!

Way back in May when the subject of client-server was first raised in this column, I said I'd cover the background and then go on to look at how much work was actually involved in installing a server-based RDBMS and connecting to it from a client. Since then, we've looked at the hardware requirements and the pros and cons of different operating systems and RDBMS products. I came to the conclusion that NT was the operating system most people would choose. As for the RDBMS, I am aware that many people will default to SQL Server. Having had a detailed look at IBM's DB2 for NT, I consider it to be a better product overall, particularly in its most recent incarnation (version 5) which should have hit the streets by the time you read this. In addition, IBM tells me that it will be cheaper than SQL Server, which has to make it attractive.

In fact, I liked DB2 so much, I bought the company. (Not strictly true. If I had enough money to even think of buying IBM, I'd simply retire and race cars. I don't even have shares in it). However, I do like it so much that I have devoted two months of my life to writing a book about it (*see page 276*). The next section is adapted from that book, mainly reduced in wordage since its purpose isn't to tell you all about installing DB2, but rather to show that server-based RDBMSs are almost ludicrously easy to install.

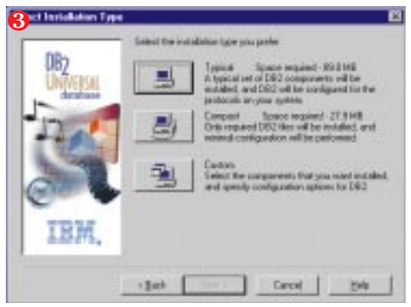
Installation

You can install DB2 on an NT server, an NT workstation and even on a Windows 95 machine. I reckon the majority of people will be installing on an NT server so that is the install we will describe, although installation on the other systems is very similar.

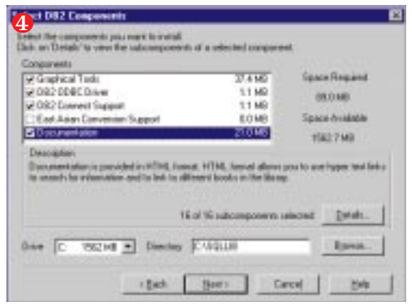
The installation of DB2 for NT will come as a revelation to those used to the standard IBM installation routines. It is no longer a long-winded, unwieldy process stuffed full of deeply searching questions and guaranteed to take up days of your precious time. Instead, it is just like that of any other NT program; you are guided by dialog boxes with helpful information and default settings.

So, where do you start? Strangely, you start by ensuring that you have a user account with NT administrator privileges and a user name with eight or fewer characters. Yes, I know this sounds weird, but user names in DB2 have to be eight or fewer characters in order to provide compatibility with DB2 running on other systems.

There are other restrictions besides the one on length: for example, you shouldn't use the characters / or ~, but if you stick to the usual ones (including @ # $ and _) you should be fine. Since the install process automatically registers the installer as a DB2 administrator, the NT name of that installer has to be restricted in this way. (In case you now begin to wonder, as I did, whether this means that all NT users who use DB2 have to have short names, the answer is "no"). The easiest thing to do is to create a new NT user with a suitably short name and assign administrator privileges to that user. With that done, you can start the installation.



1 This tells you how wonderful DB2 is and how wonderful you are for using it. Click the Next button. A box says: "Space requirements are being calculated" so select the product you want to install. 2 What you install will depend upon what you've bought from IBM. Choose what sort of installation you want: typical, compact, or custom. 3 With each one comes an idea of the space required and a brief description. Custom is the recommended option as it allows you to see more of what's going on. Select everything that looks appropriate. If in doubt, select it. It *can* be added later, but you've got to find the CD, shut down DB2, re-run the install etc.



4 When asked whether to autostart the Control Center, reply "yes" unless you have reasons to choose otherwise. 5 Choose a folder for DB2; the default suggestion of DB2 for Win NT is a good one. 6 Next, the communications protocols. Accept the defaults at first. Supply a user name and password to enable the Administration Server to log on to the system. This user must have administrator privileges, so enter the details of the user name and password that you're using presently for this installation. Once you're happy, click on the Finish button and the install should proceed smoothly. Finally, reboot the machine. That's the server installation complete.

Login to the NT server as the user you have just created. You are going to have to re-boot the server, so make sure you are the only user on the machine. Close all programs that are running (including those that might have autostarted without you noticing) and put the CD-ROM into the drive. The install program will then start up and you'll see a Welcome dialog on a big blue background.

Access Tips & Tricks

■ Type conversion

If you convert a number field into a Yes/No type, Access will convert all null values and all non-zero values into No. All other values are converted into Yes.

■ Removing primary keys

Assigning a primary key during table design is easy: you just use the key button. Removing a primary key is slightly more devious: you have to call up the Indexes box (using the fifth button from the left on the toolbar), highlight the entry for the primary key and press the delete key on the keyboard.

■ Stars in your eyes

Setting the input property for a field to "Password" causes the data to be displayed as a series of stars, rather than as values.

■ Text field size

Don't bother setting the maximum number of characters for a text data-type in Access in order to save storage space: it doesn't. You may have been fooled into thinking that it does (I certainly was) because several authoritative books make this claim (including the excellent *Access 97 Bible* that I reviewed a couple of months ago).

I began to suspect that it wasn't true, so I checked with Bill Marklyn (see book reviews, page 276). He says that Access dynamically allocates storage space to text data-types, so no disk space is saved whether you choose five or 255 characters for the size. All your choice affects is the number of characters that the user is allowed to enter into the field. (However, choosing the correct numeric type (Byte, Integer, Long Integer) does matter in terms of storage space.)

p276 ➢

Become a groupie!

Join the Access User Group. User groups are fun because they are run by enthusiasts rather than marketing droids in suits. The AUG is no exception, and the meeting I attended recently was stuffed full of useful information. Not only did it afford the pleasure of talking to like-minded enthusiasts, it also included talks on, for example, the pleasures (and pains) associated with moving Access applications to SQL Server.

The next major event is a national seminar to be held at Microsoft's UK HQ in Wokingham. Individual membership costs £65 (plus VAT).

For more details contact UK Access User Group, Stokesley House, 53 Prestbury Road, Cheltenham GL2 2BY. Tel 01242 256549. Fax 01242 226021.

What do *you* think?

As always, feedback from readers is very welcome, particularly with regard to the client-server products you are using or thinking about using. If the majority of you want to talk about Microsoft's SQL Server, that's fine — but I need to hear your thoughts and needs. You never know, I might write a book about SQL Server, too...

More on Access currency formats

Two issues ago I said: "If you choose currency as data-type for a field, Access will use whatever Windows has been told is the default currency (via the control centre). However, once a data-type has been assigned to be, say, dollars, Access *won't* change this to pounds if the data file is moved to a Windows machine where the default currency is pounds." I've been playing with this over the last month and have discovered that you can control currency formats more precisely, if you wish.

When you choose a currency format during table design, it can either be declared generically as "Currency", or explicitly as, say,

```
#,##0.00" mk";-#,##0.00" mk"
```

(a format I get if I tell my NT machine that I am Finland). The former format (if you see what I mean) will convert to the local currency as the table is moved between machines, and the latter won't. During table design, you can set the format manually to either of these as you see fit.

And on the bookshelf this month...

OK, so it *is* really weird for an author to review his own books, but, having just had two published, one of which is being promoted by *Personal Computer World*, it seems to me to be overly self-effacing to ignore them completely.

Inside the Relational Model – with examples in Access
Authors Mark Whitehorn & Bill Marklyn
Publisher Springer-Verlag
ISBN 3-540-76092-X
Price £19.50; soft cover, accompanied by a CD-ROM of examples and samples
As the title suggests, this one is about the relational model. I believe that, in order to build good relational databases, you need to understand the relational database model. Many of the textbooks that I have on the topic are authoritative but, sadly, lacking in realistic examples and (dare I say it) a sense of humour.

The subjects covered in the book have been carried in the *Hands On Databases* column over the past four years and this book draws that information together in an integrated way. There is also much new information, as well as a CD-ROM with all the sample files.

The audience at which it is aimed includes those who are building a database and who want to know more about the process in order to reduce the potential for encountering pitfalls later. The section on SQL is also a useful primer for anyone new to the language.

DB2 for Windows NT – Fast
Authors Mark Whitehorn & Mary Whitehorn
Publisher Springer-Verlag
ISBN 3-540-76200-0
Price £19.50: soft cover, accompanied by a CD-ROM
The focus of the column has, by popular demand, been turning towards the topic of client-server databases, and the second book is a "get-you-up-and-running-quickly" guide to IBM's latest client-server product, DB2 for Windows NT. I have always been impressed by IBM's database engine (a robust, elegant and speedy implementation, just like a database engine should be) though dismayed at the tenacity with which the company clings to its command-line interface. This latest version of DB2 has a GUI; yes, a proper, user-friendly GUI with all mod cons.

PCW Contact

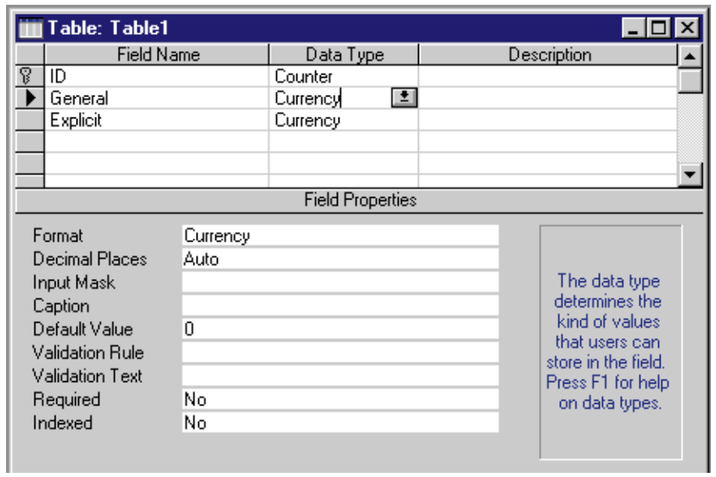
Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column. Write to him at the usual *PCW* address or email database@pcw.co.uk.

Primary purpose

Mark Whitehorn attends to the use of primary keys in data tables. You can also read about graded data and how to get into the currency control business with Access.

Last month I wrote: "If you choose currency as data-type for a field, Access will use whatever Windows has been told is the default currency (via the control centre). However, once a data-type has been assigned to be, say, dollars, Access *won't* change this to pounds if the data file is moved to a Windows machine where the default currency is pounds." I've been playing with this during the past month and have discovered that you can control currency formats more precisely if you wish.

When you choose a currency format during table design, it can either be declared generically as "Currency" (Fig 1) or explicitly as, say, #,##0.00" mk";-#,##0.00" mk" (Fig 2) — a format I get if I tell my NT machine that I am Finland. The former format (if you see what I mean) will convert to the local currency as the table is moved between machines, and the latter won't. During table design, you can set the format manually to either of these as you see fit.



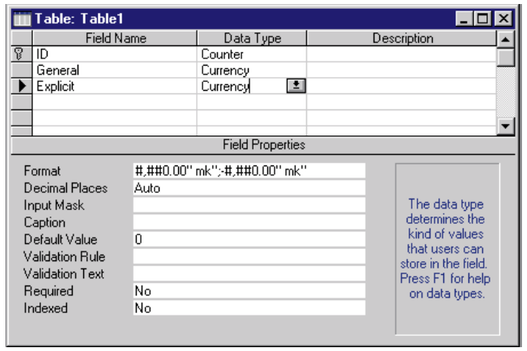


Fig 1 (above) Declaring a currency format generically during table design...
... and declaring it explicitly (Fig 2, left)

Grades

Some data inherently cries out to be graded. OK, I'm thinking about student marks here: 70 or above = A, 65 - 69 =B and so on. But you might be a travel agent thinking: Cost £500 or above = Luxury, Cost £400 - £499 = Business Class… down to, Cost less than £50 = Economy.

Whatever you need to grade, always store the value to be graded (mark, cost and so on), never the grade. You can use a query that looks at the original data and calculates the grade you need. I have included an MDB file called UOD on our cover-mounted CD-ROM this month. This stores students' examination marks in a table called ATTEND (that also stores which student attends which course). The query called Grades uses a formula:

```
Grade:IIf([Exam]>=70,"A",IIf([Exam]
>=65,"B",IIf([Exam]>=60,"C",IIf
([Exam]>=50,"D",IIf([Exam]>=40,"E",
"F")))))
```

to assign a grade (Fig 3). The formula may look horrible at first sight but it is simply a nested IIF statement. At its most basic it looks like this:

```
Grade:IIf([Exam]>=70,"A","B")
```

which says that if the value in Exam is greater than or equal to 70, assign an "A", otherwise assign a "B".

Database cure

"*I am writing an occupational health database. Each nominal*" [I wonder what a nominal is? — MW] "*may have more than one previous occupation, stored in the table 'tblMinPrevOcc'* (Fig 4).

"*I want to run a query which will tell me how many nominals have had a previous occupation: i.e. how many unique values are there in the fldRefNo field? I thought I could do this with the following statement:*

```
SELECT COUNT(DISTINCT fldRefNo) AS
fldPrevOccCnt
FROM tblMinPrevOcc;
```

but I get a syntax error.

"*I feel that the design of the database may be at fault but am not sure what is the best solution. I am using Access 2.0. Any suggestions?*"

Mark Capaldi

The design is certainly worth examining, since there is no obvious primary key for the table tblMinPrevOcc. You could use all three fields, but that is getting slightly messy and you would then exclude the possibility that the same person had, on more than one occasion, done the same job for the same length of time, which is unlikely, I know, but still possible. A better alternative might be to add a Counter field (but see "Candidate keys", later).

However, I digress. Here is a solution to your problem. Create a query called (for the sake of argument) "Unique" which is as follows:

```
SELECT DISTINCT tblMinPrevOcc.
FldRefNo AS fldPrevOccCntFROM
tblMinPrevOcc;
```

which is very similar to your original. This shows (but does not count) the unique entries in the table:

FldPrevOccCnt
10003
10230
10458

You can then base a query such as

```
SELECT DISTINCTROW Count(Unique.
fldPrevOccCnt) AS [Total No Of
Nominals]FROM Unique;
```

on this query called UNIQUE. This second query will return the number you want:

Total No Of Nominals
3

A sample is on our cover-mounted disc this month as NOMINALS.MDB.

Candidate keys

Mark Capaldi's question (*see "Database cure"*) obliquely raises another: given that all tables should have a primary key, how do you choose the field, or combination of fields, to use as the primary key?

The answer, delightfully, is often not as simple as it seems. "What do you mean, it isn't delightful?" I hear you cry. If all the decisions we had to make during database design were easy, where would be the fun? And, more to the point, how could we justify our huge salaries?

p248 ➢

	FIRSTNAME	LASTNAME	COURSENAME	Exam	Grade
	MIKE	WELLINGTON	Cytogenetics	76	A
	MIKE	WELLINGTON	Intro. to Polymorphism	67	B
▶	MIKE	WELLINGTON	Automotive Design	72	A
	SALLY	JONES	Simple Ergonomics	56	D
	MIKE	WELLINGTON	Ecology	23	F
	MIKE	WELLINGTON	Population Genetics	45	E
*					

Fig 3 (above) Assigning a grade

Fig 4: blMinPrevOcc

FldRefNo	PrevOccCode	LenPrevOcc
10003	1234	4
10230	4567	2
10003	7896	5
10458	1256	2
....	....	..
10230	5693	1

Fig 5: Customers; but which candidate key?

CustomerName	Address	ManagingDirector	TelephoneNo
Fred Bloggs Ltd.	23 Park Rd, London	Brian Bloggs	098 765 432101234
Helen Shipping	46 Troy Lane, Paris	Helen Oftroy	01234 567 8901233

Most people (he wrote hopefully) know that a primary key is a field or combination of fields, the values which uniquely identify the records in the table.

Let's assume that you work for a company called PenguinShipping. This company manufactures items which it then sells to customers. As the database designer, you decide that one table for CUSTOMERS and another for ORDERS is appropriate.

ORDERS

OrderNo.	Value	ShippingDate
1	£50	1/9/97
2	£75	2/11/97

Assuming that each entry in OrderNo is different, that field is an obvious candidate to be the primary key because the value therein uniquely identifies the record to which it belongs. I use the word "candidate" advisedly in this case since this is actually the technical term used to describe any field, or combination of fields, which could be a primary key.

Clearly there are other candidate keys in this table. We know that a primary key can be made up from multiple fields, so given that OrderNo already uniquely identifies the records, OrderNo + Customer must also be a candidate key. By the same token, OrderNo + ShippingDate is also a candidate key, but Customer + ShippingDate is not a candidate (unless we happen to know that two or more orders are never shipped to the same customer on the same day).

However, it is often impossible to identify candidate keys simply by studying the data. Additionally, you often need to have an understanding of what the data means, how it is used, and what are the company rules for that data.

For example, suppose I tell you that PenguinShipping actually makes ships and that the Value field is in £m. ShippingDate is actually the date on which any given ship is launched. If I also tell you that the port authorities forbid more than one launch per month, you can suddenly identify ShippingDate itself as a candidate key, since there can never be more than one record with the same ShippingDate.

Clearly, once you have identified the candidate keys in a table you have to pick one of them to be the primary key. As it stands, OrderNo is almost certainly the most reasonable candidate in the table above, but consider the one shown in Fig 5.

Suppose that all customer names are guaranteed to be unique, as are all addresses and telephone numbers. Here we have a plethora of candidate keys. Not only can we use any one of those three, we can use any combination of any keys except ManagingDirector on its own. Assuming that common sense prevails, the most obvious choice looks like CustomerName.

This means that the ORDERS table can now use a foreign key to reference the CustomerName field as in Fig 6, and all is well. But, suppose you now build exactly the same type of database, only this time you work for PenguinConfections. This company doesn't ship ships, it sells sweets. In this case, electing to use the CustomerName field is less likely to be satisfactory. Which raises the question, why should the product make any difference to the database design?

The answer lies in the number of records in the ORDERS table. PenguinShips will be lucky to complete ten ships a year. This means that the ORDERS table will remain tiny. PenguinConfections, we hope, will ship many more orders than this. Assuming that it ships, say, 10,000 orders per year, then the ORDERS table will hold 10,000 records. Imagine all of those customers' names written dozens of times each — wasting space, cluttering up the disk and slowing down the database. The obvious answer is to add a numeric field to CUSTOMERS and use that as the primary key (see Fig 7).

This is a great solution (and one that many people would employ almost instinctively) even though it renders the ORDERS table less readable, since we now have a number rather than a name. However, it is worth remembering that this solution brings its own overheads because you have added a field to CUSTOMERS.

Under certain circumstances this might be a mistake. Suppose that your CUSTOMER table is mega-huge (because it stores all your potential customers) and even though your ORDERS table has 10,000 entries it only actually references a small proportion of the records in the CUSTOMER table. Adding a CustomerNo field to CUSTOMERS will reduce the size of the ORDERS table but the concomitant increase in the size of CUSTOMER may negate the gain.

So it is yet another case of horses for courses. In practice, I tend to do the following: if a table has an obvious, relatively short, single field candidate key then I use that as the primary key. This is not uncommon (Part Number, Student matriculation number, National Insurance number and so on). If such a field is not obvious, by default I tend to add an artificial numbering field such as Counter/ Autonumber in Access.

Client-Server

I haven't forgotten that this is a continuing topic, but there is not enough space this month. I will return to it next time, bringing laughter and song.

Fig 6: Orders

OrderNo.	CustomerName	Value	ShippingDate
1	Fred Bloggs Ltd.	£50	1/9/97
2	Helen Shipping	£75	2/11/97

Fig 7: Adding a numeric field

CUSTOMERS

CustomerName	Address	ManagingDirector	TelephoneNo
Sam's Sweets	16 Green La., London	Sam Sugdon	0123432101233454
Bon Bons	12 Apple Lane, Cheapstow	Billy Brown	0143210987654342

ORDERS

OrderNo.	CustomerName	Value	ShippingDate
1	Sam's Sweets	£24	1/9/97
2	Bon Bons	£45	1/9/97

PCW Contacts

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column. Write to him at the usual *PCW* address or email database@pcw.vnu.co.uk.

UK Access User Group, Stokesley House, 53 Prestbury Road, Cheltenham GL2 2BY. Phone 01242 256549; fax 01242 226021

And that's magic...

Mark Whitehorn conjures up tricks to use in Access. He finds it is a product that knows where its towel is, without any flannel, and has meta-data which needs to be stored.

You can use Shift F2 virtually anywhere in Access. It opens a "Zoom box" where you can edit the text upon which your edit cursor was sitting. Use it whenever the text you are trying to read sits in a small box. It is mega-useful when entering a complex condition in a query, say. It works for users of your application who can be encouraged to use Shift F2 when entering data into a form.

You can also use Shift F2 while programming. If you place the edit cursor on a user-defined function, pressing the magic key combination will jump you to the definition of that function. Doing the same on an Access function like CurrentDB (i.e. not user-defined) will open up the object browser at the appropriate place.

Committing records

Another useful tip to feed to your users is that Shift Enter will commit an edited record to disk without the need to move onto the next record. (Or click on the edit record symbol on the left of the form.)

Fig 1 **Inside the data dictionary. Intuitive, isn't it?**

Access currency formats

If you choose currency as data-type for a field, Access will use whatever Windows has been told is the default currency (via the control centre). However, once a data-type has been assigned to be dollars, say, Access won't change this to pounds if the data file is moved to a Windows machine where the default currency is pounds. This makes perfect sense (since the absolute value of items would alter as data files were moved from country to country) but needs to be remembered if you are distributing an application into other countries.

A free toy for data mining in the data dictionary

Access is a product that really knows where its towel is. Well, in the literal sense it has no idea of the physical location of its textile de-humidifying implement; in fact, it probably hasn't got one. What it does have is the database equivalent, which is called meta-data, and it certainly knows where that is — in the data dictionary. Your problem is trying to read that data and I have a solution.

A database consists of the user's data stored in tables. A DBMS also has to store information which "describes" the database itself (the names of the tables, the names of the queries, the explicit joins between the tables, the data integrity constraints etc). Such information is known as meta-data. One of Codd's rules that define a Relational DBMS says that the meta-data can't be stored any old way; it must be stored as data in tables, just like the user's data. Access obeys this rule and stores the meta-data in a series of hidden tables, the names of which all begin with MSys (presumably standing for Microsoft System). The data in the tables is the meta-data, the tables themselves can be referred to as the data dictionary.

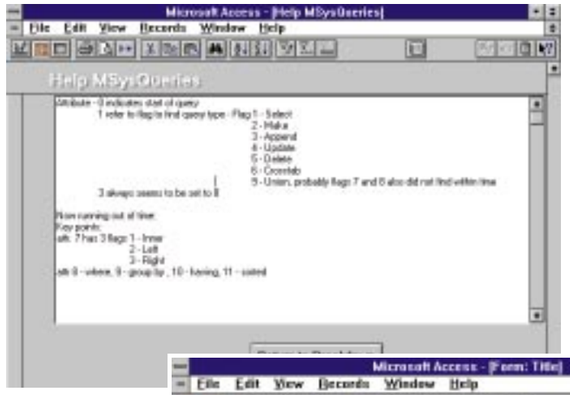
Codd says that storing the meta-data in the same way as the user data ensures that the users of the database can access the meta-data in the same way in which they access their normal data. So any user who has learnt to query their data to find out, say, how many customers live in Hereford, can use the same techniques to find out how many forms there are in the database.

All of this is fine and dandy in theory but in practice it rarely works, for a very simple reason: the database designers are free to use whatever data format they like within the tables to represent the structure of the database, and they seem to choose very abstract formats. In Access, for example, data about the queries is stored in a table called MsysQueries (Fig 1).

The first 15 rows in the table shown, define one query in the database. In order to find out what sort of query it is, you have to find the value 1 in the Attribute field within the first 15 rows and then look up the value in the field called Flag.

Flag value	Meaning
1	Select
2	Make
3	Append
4	Update
5	Delete
6	Crosstab
9	Union

p266 ➢



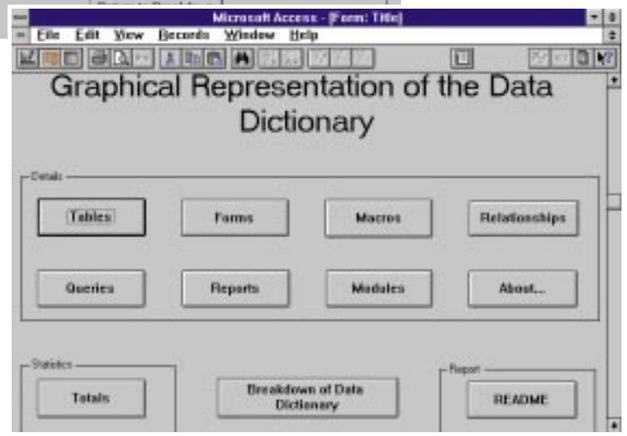


Fig 2 (left) Some of the useful information inside Black.mdb

Fig 3 (below) The GUI front-end to Black.mdb

I don't know about you, but I couldn't have worked that out from a quick glance. None of this is documented (at least for normal mortals) so how did I find out? For my sins, I teach the database course at Dundee University. One of the projects I set the students this year was to build an Access application that displayed the meta-data in a user-friendly fashion. In order to do this, the students had to "reverse engineer" the data they observed in the data dictionary, working out what all the bizarre formats meant before generating queries that made sense of them.

Student demonstration

It seemed a shame for this work to be wasted, so I am including one of the better ones on our cover-mounted CD-ROM (in Black.mdb). This was written by one Johnny Black and includes some notes he made about the formats used within the data dictionary.

Three riders: first, remember this was a student project, so neither Johnny nor I can guarantee that all his conclusions are correct. Second, the structure of the data dictionary varies between different versions of Access. Black.mdb was written specifically for version 2.0 and may well not provide accurate information if imported into later versions. Third, this work was completed to a tight time schedule and had I given the students more time could obviously have been improved. With those riders, Black.mdb should still serve as an excellent starting point for anyone who wants to examine the data dictionary.

Operating theatre

Having covered the hardware aspect of client–server computing, we can move on to a more contentious area: the operating system and the back-end DBMS.

Assuming for a moment that you do not want to stray into the realms of mainframes just yet, the obvious operating systems onto which a PC database application could be up-sized are UNIX, Novell NetWare and Windows NT.

UNIX is possible, but my experience is that most PC users tend to shy away from it, which is a shame because it's a very stable platform for RDBMSs.

Next comes NetWare, which some people may consider, especially if they are using it as their NOS for file and print. The problem is that NetWare was optimised for precisely those functions, not as an application server. So although products like Oracle 7 are available for NetWare, in practice they sell like cold cakes.

I love NetWare dearly as a NOS but cannot recommend it as an operating system for RDBMSs.

Finally there is Windows NT. This is not as stable as UNIX, and we could debate for some time whether it is really a stable enough platform for a mission-critical RDBMS. It is apparent however that many people are migrating to it and that the major RDBMS firms are providing products for it. If you choose to use Windows NT, at least you will have lots of company, so that's my general recommendation unless you have good reason to choose otherwise.

Each of the big three has an offering for NT: IBM has DB2 for NT, Oracle has Oracle for NT, and Microsoft has SQL Server. Two of these products have a history of running on other platforms, and all three have perceived strengths and weaknesses.

DB2 is famous for running on mainframes, being very big, powerful and secure, and being a pig to drive. Oracle is famous for running on UNIX, being very big, powerful and secure, and being a pig to drive. SQL Server is famous for running on NT and having a great user interface.

If you are currently an Access user, the obvious choice is surely SQL Server from Microsoft. After all, you are already happy with Access: SQL Server comes from the same company so you can expect the same quality of software. You can also expect a high level of compatibility between the front-end and the back, so is there even a question here?

The answer is "Yes", there is a serious question, but a relatively complex one with several facets. Access and SQL Server were both developed within Microsoft but by different development teams. My experience is that the teams can produce differing standards of products, so the fact that both of these come from Microsoft is largely irrelevant. Indeed, SQL Server was originally a Sybase product and still has large chunks of Sybase code buried within it which has no connection with Microsoft at all. This code includes the query-optimiser which is "less-than-optimal".

In my June issue column I suggested that many client-server databases are also mission-critical to a greater or lesser extent. There is a huge gulf between a word processor and a database which stores information crucial to the survival of a company. It generally doesn't matter much if the former falls over occasionally: the latter must be much more stable and, if it fails, must fail gracefully so it can continue to work when restarted.

Companies like Oracle and IBM have always worked in mission-critical environments and understand that, in this environment, the stability and data-integrity of the product is of paramount importance. They understand that software mustn't be shipped until it is really rock-solid.

Now ask yourself what you think about Microsoft's track record in producing software that never falls over? Ask yourself how many times you have heard about bugs turning up in new releases of Microsoft products which will "be fixed in the next version". I'm not suggesting that Microsoft can't learn about mission-critical software. What I am suggesting is that I want to see a good track record before I commit my data to a given product. I believe IBM and Oracle have already demonstrated such a track record.

"Ah, but what about drive-ability?" you ask. "You're really suggesting that we use a product with a Neolithic user-interface." (Those of you who haven't experienced the joy of playing with the current version of DB2 for NT may be surprised to discover that it still has a command-line interface. No, I'm not joking, or exaggerating. Yes, I do mean you don't get cute little buttons, icons and slider bars. You have to type a string of characters into a textbox in order to drive it.)

But no, I wouldn't inflict a command-line on anyone. The good news is that I have been playing with the new version of DB2 Universal Database for NT (due for release in October) which has a really intuitive, easy-to-use interface. In fact, it closely resembles Access (Fig 4).

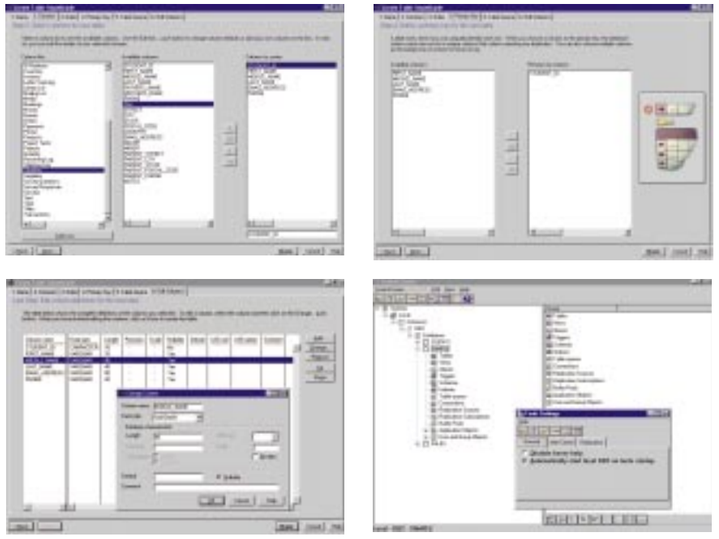


Fig 4 Using DB2 for NT is actually a pleasant experience, at least in the new version. Clockwise (from top left): Choosing fields for a table; setting the primary key; setting data types; the main control area for DB2

Of course, if we are allowed to talk about next versions, Microsoft is likely to say the next version of SQL Server will fix any current problems that exist within the product. Apart from considerations about track records in this area, a good question to ask here is, "Which company has the easiest job?" IBM, which has to bolt on a respectable interface to an already mission-critical piece of software, or Microsoft, which has to convince us that SQL Server is of mission-critical quality.

Scaling up

Scalability is loosely defined as: "How much the performance of a given product alters as conditions are changed."

A good example is the number of users attached to an RDBMS. Suppose you have an RDBMS running with 20 concurrent users. If it were perfectly "scalable", doubling the number of users would no more than double the response time for each given user. But life isn't usually like that, and doubling the number of users often makes the response time more than double for each user (because the RDBMS spends proportionately more time swapping between the users). So no-one expects perfect scalability but it is possible to compare products and see which scale most efficiently. Bloor Research has recently published a report called *The Realities of Scalability* which examines the scalability of DB2 for NT and SQL Server (it also looks at the scalability of DB2 for AIX).

The scalability of SQL Server comes in for criticism: "Put simply, Microsoft SQL Server for Windows NT at high numbers of users performs dramatically worse than either of the other two databases." As for the stability: "This database had a number of failure states that could repeatably [*sic*] be generated. With large numbers of users, it was found that the database would grind to a halt."

DB2 for NT, as you might expect given its history, does not come in for such criticism: "Despite the occasional unexplained server or database crash, it proved impossible to consistently generate a fatal error in DB2 for Windows NT. It seemed to be capable of taking most things thrown at it."

Perspective

All of this has to be seen in perspective. Suppose you work for a small to medium-sized enterprise. You have developed an Access database on a standalone PC and it has been riotously successful. Now your boss wants you to upgrade it so that ten people can use it. There is no suggestion, either from the Bloor report, or from me, that SQL Server would be inappropriate. However, I can see no reason why the new DB2 wouldn't be better and it is actually cheaper, so it is well worth considering.

Now suppose you are developing an application that may become mission-critical in the future, or which needs to support many users (say >1,000) and/or lots of data (say >100Gb). Under these circumstances, for the reasons given above, I would strongly recommend that you look carefully at the alternatives to SQL Server rather than view it as the default option.

PCW Contacts

Mark Whitehorn welcomes readers' correspondence and ideas for the Database column. Write to him at the usual *PCW* address or email database@pcw.vnu.co.uk.

Bloor Research 01908 373311

Classy chassis

Mark Whitehorn turns his attention to "professional" quality hardware for client server databases and what you can expect for your money. Plus, tips and tricks for Access.

Last month I was trying to "sell" you a very expensive piece of kit. I hoped to convince you that standard PC chassis were inappropriate as servers for mission-critical databases and that you ought to buy something expensive from a reputable manufacturer.

A fair question here is, "What do I get for my money?" In other words, a "professional" server is likely to cost twice as much as a PC of the same specification, so where does the money go? The following list is not exhaustive but it gives some idea of the extra hardware features you can expect from a "professional" server:

- RAID disk arrays.
- Fault monitoring and prediction on hard drives, processors and memory (anyone who remembers HAL at work in 2001 — A Space Odyssey will appreciate this).
- Pre-failure warranty, which allows the components identified as liable to failure, to be replaced before they do so.
- Dual peer PCI buses for high I/O bandwidth.
- Array controllers which allow RAID storage to be added while the system is in operation.
- Tape backup systems which provide rapid backup (40Gb per hour).
- Redundant power supply and UPS options.
- Redundant controllers on standby in case of failure.

Most of these are self explanatory. However, a brief word about RAID (Redundant Arrays of Inexpensive Disks) may be helpful. "Brief" is the operative word, as there are several flavours and the area is relatively complex. A RAID allows you to store your data on several disks, which holds a number of advantages as far as database servers are concerned. For a start, a RAID can be set up in such a way that if even one of the disks fails, your data remains safe because each piece is stored redundantly on more than one disk.

Clearly, this has the disadvantage that although you might have five 2Gb disks and therefore 10Gb of disk space, you won't be able to store 10Gb of data. The data is more secure, though. Additionally, there is a rather elegant spin-off from the redundant nature of RAID which is that the data can often be retrieved more rapidly.

Furthermore, RAID can be hot-swappable, so if a disk fails you can extract the duff one, slot in a good one and the data which was on the failed disk will be reconstructed from the others. The database can continue to perform, albeit more slowly, until the new disk is fully online.

It doesn't take a genius to work out that the money is going into two main areas: speed, and keeping the server up and running. You may never need the RAID array; that redundant controller may spend all its time idling. Nevertheless, it would make your day having arrived one morning to discover that even though one of the disks has failed, your system is still running.

Remember it well

A database server needs RAM, and then it needs more RAM. A database server will use RAM to hold the RDBMS engine and to cache data, transactions and indices. The more RAM you give it (within reason) the faster it will work. Think in terms of 64Mb as a minimum and 100Mb as a more reasonable starting point. Make sure the server you buy has free memory slots so the

Time, gentlemen, please…

Following reader Gareth Wade's query about adding time lengths to give a running total, (*Quickies, PCW* April) I have since received a couple of solutions from others.

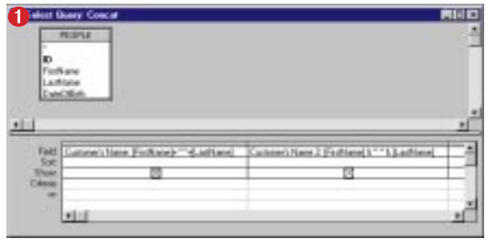
■ MA Roberts writes: "*Regarding the 'Database Quickie' from Gareth Wade (PCW April) I have had a similar problem to Gareth's when adding together time values. The short time format field in Access does not work satisfactorily for adding values that will exceed 23:59. One workaround I am using is to convert the time field into a number type. When adding time records together in a number-type field, Access converts the time to a fraction of the 24-hour day. It is therefore necessary to multiply this field by 24 to get a value for hours and decimal minutes. This field may then be used to obtain a value for time (in hours and decimal minutes) multiplied by an hourly rate such as might be used in a time-sheet style application.*"

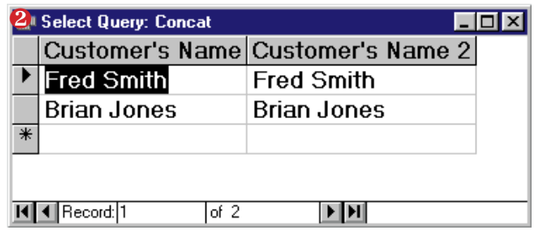
■ And Vidar Eggen writes from Oslo: "*I had the same problem when wanting to report on the added flying time of pilots as well as aircraft at my flying club. What we really need in Access is the [ttt:mm] format of Excel. Until then, I use this basic formula:*

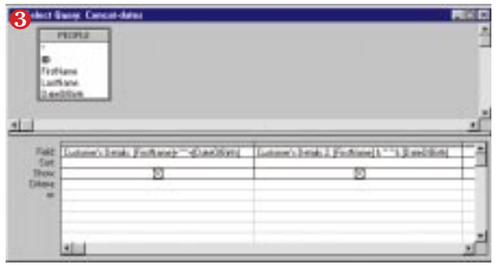
```
=Str$(Sum(Hour([Time]))+Sum(Minute([Time]))\60) & ":" &
Format$(Str$((Sum(Minute([Time]))) Mod 60);"00")
```

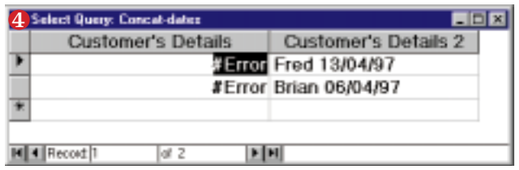
"*The output of this formula will be '123:45' in text format. The reason for converting the sums to a string is to make sure that five minutes is printed as '05', not '5'. If quoting from this, please rephrase my language into readable English!*"
(Vidar's English seems fine to me and is considerably better than my Norwegian! — *MW*)

p284 ➢









memory can be expanded later without replacing all the RAM it currently holds.

Purchasing processors

In general, buy several processors. Simple operating systems can only use a single processor, but complex ones like UNIX, VMS and NT4 Server can use multiple processors. Mind you, just because an operating system can make use of several CPUs doesn't mean that the programs which run on top of it can too. But all the respectable RDBMSs can and do, and this isn't restricted to client-server RDBMSs.

I recently mounted Access 97 on an NT4 machine and turned it loose on a large database. Rather to my surprise Access used all the four processors on the machine and was able to query 1,000,000 indexed records in a couple of seconds. Interestingly, Excel used only one of the available processors and its performance was startlingly unimpressive.

So the bottom line is, what do I use as my test database server? I am currently running a Compaq ProLiant 5000 with four 166MHz Pentiums, 384Mb of RAM and 9Gb of disk space. I look upon it not so much as a server, more as a non-intrusive sleeping aid. Okay, it is more expensive than the tablets, but it has fewer side effects.

Listing 1

```
"Report generated " & Now()  & " for " & [Customer]
```

Listing 2

```
Between Date()-7
And Date()
```

Listing 3

```
Between #1/1/1981# And Date()
```

Tips and tricks for Access

By popular demand, I am introducing tips and tricks this month. It is aimed at Access users and will include information which ranges from simple to complex and should provide something for everyone.

1. **Queries**, by default, use the field name as the column name in the answer table. If you ever want to alter this, insert the text you would prefer to use, followed by a colon, before the field name. Thus:

```
Customer's Name: [FirstName]
```

will replace "FirstName" with "Customer's Name" in the answer table. Which leads us neatly to the second tip.

2. **The "+" operator** can be used to concatenate text strings, as many people know. Thus:

```
[FirstName]+[LastName]
```

in a query will join the two strings together. But this unhelpfully yields names like "BrianJones". So, simply add in a literal space like this:

```
[FirstName] + " " + [LastName]
```

which then gives: "Brian Jones".

However, you can also consider using the lesser known "&" operator. Amazingly,

```
[FirstName] & " " & [LastName]
```

gives exactly the same result: "Brian Jones" (Figs 1 & 2). You're thinking, "Hello, he's flipped. If it gives the same result, why bother to use it?" Well, the advantage of getting into the habit of using "&" is that it automatically converts all data types into text. This has no effect if the two fields are already text (as in the case above) but it does wonders with expressions like

```
[FirstName] & " " & [DateOfBirth]
```

which will produce "Brian 06/04/1967" rather than the "#Error" produced by the + operator when it tries to concatenate data from two disparate data types (Figs 3 & 4). In other words, if you habitually use "&" instead of "+" you won't have to use type conversion functions.You could even use it in a report like Listing 1.

3. **When adding fields to a query grid**, you can drag and drop them as usual. However, you can also select multiple contiguous fields using shift-click on the first and shift-click on the last, and then dragging and dropping as usual. You can also add non-contiguous fields by CTRL-clicking on them.

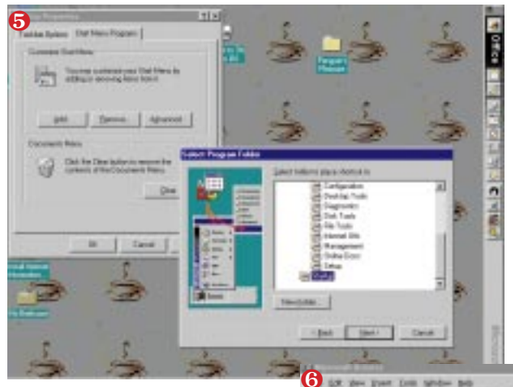
4. **When looking at a table of data**, you can resize any field to fit the widest data in the field by double-clicking on the right-hand column border (this has to be done at the top of the table). You can hide the column by dragging the right-hand column border across to the left-hand border. You *can* open this up again with the mouse but it takes practice, as you often simply widen the adjacent column instead. It is often easier to use the menu system Format

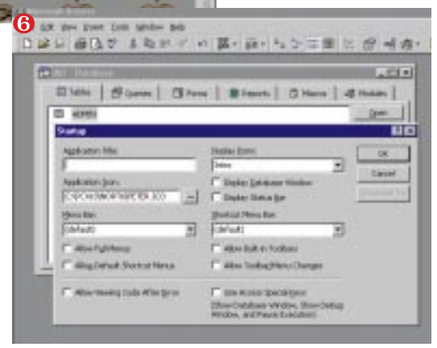
Listing 4

```
Set button OnClick event to "=Clicked([Screen].[ActiveControl].[Caption])"
```

Listing 5

```
Function Clicked(capStr as string)
    Debug.Print capStr
'Or assign it to the parameter for the query
End Function
```





Show Columns, or Format Unhide Columns, in Access 97.

5. **Record selector.** Most of us have discovered the delights of the "Find Specified Text" button (the binoculars) which will search a table for information. By default, this will search the current field and the dialog box allows you to specify the entire record. However, it is often easier to click the record selector (at the left of the record) before starting the search and this automatically sets the search to run against all fields. This works in form views as well as in datasheet views.

6. **"Between… And".** As regular readers will know, I am obsessed with the idea that the data in our databases should be as "clean" as possible. Take something as simple as a DateOfBirth field in a maternity database. The default (for the child, *not* the mother) might be the function Date().

But perhaps not every child's record might be entered on the day (staffing levels being what they are) so the simple answer is to allow any entry. But that opens the door to really whacky entries. Suppose we know that all entries, no matter how delayed, are always brought up to date on a Sunday, say. This means that the longest delay would be seven days, so the default could remain as Date() and the validation rule could be set as Listing 2. The "Between… And" operator is a useful one to get to know. You can use it with fixed dates like Listing 3 or for data types other than dates:

```
Between 1 And 45
```

Gaining automatic Access

A reader has written in for help: "*I'm struggling with Access 97 for NT, trying to build a database for other people to use. I have seen other databases where a form opens up automatically, which is what I want but how do I do it?*

"*Additionally, since many of my users are not used to databases, is there any way of limiting the options which are available to them when Access is running?*"

"*Finally, is there a way of getting Access to start as soon as NT fires up? Can I set up the machine so that whenever the user switches on the PC, it goes into NT, lets them log in, and then goes into Access?*"

The simple answer is yes. Let's start from the outside and work inwards. To fire up a copy of Access, you can add it to NT's StartUp menu. Click on Start and choose Settings, TaskBar and the Start Menu Programs tab. Click Add and then browse in the normal way to find Access. Then click on Next, select the StartUp folder and finish. This seems *more* complex than it used to be in Windows 3.1 — weren't operating systems supposed to be getting easier to use? (Fig 5)

Access 97 has a new startup form which allows you to control many options when it kicks into life. You can get to this by right-clicking the border of the Database window and choosing StartUp, or go via the menu with Tools, StartUp. Fig 6 shows the sort of options that are available.

The subject of passing captions from buttons has exercised the brains of several readers. For instance: "*I was intrigued by the buttons and captions capture question. Your method in the February issue appears to be the way to go and can be speeded up using [Listing 4] (using Access 7) which calls [Listing 5].*" **Kelvin**

Malcolm Fraser, too, sent in a solution and it's on our cover-mounted CD, this month, in a file called CAPTIONS.MDB. Grateful thanks are also due to Steve Foster, Richard Todd, Malcolm Bacchus, Wilf Davies, Alan Berry and others who provided elegant solutions.

Beyond the pale™

I discovered the following rumour on the web site of Bloor Research, at www.bloor.co.uk. It appears that the word Metadata™ has been trademarked™ to a pharmaceutical™ company in the US. At first sight this sounds (if you'll forgive the mixed metaphor) unbelievable, but I suppose those people who issue trademarks in the US may not be as familiar with planet database as those of us who live there. To make matters worse, the company that was awarded the trademark™ has started sending "Cease and Desist" letters to anybody it finds using, er, well, that word™. I'm currently trying to trademark the word "pharmaceutical™ " in the hopes of wreaking some small revenge™.

PCW Contact

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column, at database@pcw.co.uk.

Serves you right

Well, he promised, and here it is. As a new, regular section of his column, Mark Whitehorn introduces the subject of client-server computing explaining how, why and wherefore.

As promised last month, a section of this column will now be devoted to the topic of client-server computing.

There are several definitions used in client-server databases. For our purposes I will use the definitions shown below unless stated otherwise. What we can discuss over the next few months is:

1.Server Suitable hardware, OS and RDBMS.

2. Client Suitable hardware and software.

3. Component positioning Where you should place the business rules and the data (yes, I know I said it sits on the server, but there are some exceptions!).

We can also look at how an existing single-user system can be upgraded to client-server: that little lot should keep us busy for a few months.

Background

Most companies upsize to a client server because they need to change a single-user database into a multi-user. Although this can be done by moving the data onto a file server (see my previous columns), such solutions are usually limited in both power and the number of users. For many companies, the only really effective way to provide multi-user access to a database is by moving to client-server.

In many cases, the change to client server implies another subtle change: the database changes from a useful but non-essential part of the business to a mission-critical system. I am not suggesting that all multi-user databases are mission critical, simply that in my experience many become so. The following is a useful conversation to have with your boss before you start.

"Suppose that we go ahead and build this client-server database, and suppose that it is as successful as we all hope. Now imagine: once it has been in place for six months or so, it begins to fail sporadically. How dangerous will that be to our business?"

If the answer is: *"It will be annoying, but we can live with it because (insert appropriate answer here)…"*, you don't need to read the rest of this section. If the answer ranges from: *"Well, that would be difficult to quantify…"* to *"Such failures, if prolonged, would seriously damage the company"*, you need to think very carefully about hardware for the server and be prepared to spend some serious money.

Incidentally, you need to keep a lookout for coded, political answers like: *"We are sure that any system you build will be reliable."* This appears not to answer the question (hence the political reference) but in fact it does. It decodes as: *"We can't afford an unreliable system, and you will be fired if we get one. But, of course, we don't want to spend any extra money."* If you receive such an answer, as far as you and your career are concerned, this is now a fully mission-critical system.

If you think your client-server database is or will become mission critical, bear the following in mind. Most people are used to the fact that a typical PC costs around £1,000. What you have to do is make them realise that while that's fine for a word-processing machine, it's totally inappropriate for a mission-critical database server. For a start, database servers, by their very nature, work harder and need to be of a higher specification than a normal PC. For another thing, if a word-processing PC fails, those important letters can be typed up on another PC. If your database server fails, what are you going to run the company accounts on?

The nitty gritty

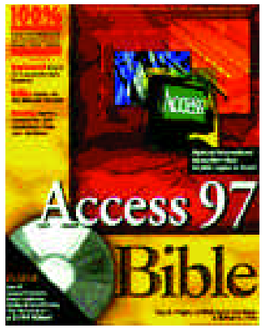
So, as you will have guessed, we have reached the part of the column where I try to get you to spend a small fortune on your server hardware. Please understand that as I try to part you from your hard-earned cash, I don't have shares in any of the hardware vendors who may be mentioned here. Rather, I just want to make sure you keep your job.

Client server definitions

- **Standalone database** — Runs on a single machine. That machine is typically a PC running Windows and an RDBMS (such as Microsoft's Access). The data and the data-processing engine all reside on this one machine, so multi-user access to the data is not possible.
- **Client server** — The data and the data-processing engine are moved across the network and run on a dedicated machine called a database server. Since multiple-client PCs can access this server, the system becomes multi-user. The clients will still typically run Windows and some sort of interface to the database.
- **Client** — A PC running Windows (of whatever flavour).
- **Server** — A computer (not necessarily Intel-based, but probably so) which runs a dedicated RDBMS back-end. This will not be Microsoft Access, since it cannot run as a database server, but instead will be something like Microsoft's SQL Server, Oracle's Oracle, or IBM's DB2. The server will also run on a server operating system such as Windows NT, OS/2 or UNIX.

p270 ➢

Book review: Access 97 Bible



Cary Prague has established himself as one of the more authoritative Americans to write about Access. His earlier books have been excellent and this one (over a thousand pages long), written in conjunction with Michael Irwin, is no exception.

It tells you how to use Access' GUI and macro language but stops short of programming in VBA (Visual Basic for Applications). Subjects are covered in detail and with accuracy. It is only when the authors stray into areas which are more to do with the relational model than Access that the content becomes a bit flaky. As an example, they attempt to distinguish between a one-to-many join and a many-to-one join. Since these are, as the authors acknowledge, essentially the same animal viewed from a different direction, it seems like needless obfuscation. It isn't helped by their statement during this rather bizarre interlude, that a many-to-one relationship is (in theory) a one-to-one relationship. Surely a misprint?

However, this minor carping on my part should not, under any circumstances, prevent you from buying this book for the treasure-chest of Access gems with which it is stuffed. For novice Access users, it will be an invaluable road-map. For experienced users, it is a wonderful source of tricks and tips. As is so often the case with books of this type, one single example or tip can save you, say, a couple of hours' work. This is a "must-have".

■ ***Access 97 Bible*** **by Cary Prague & Michael Irwin. £42.99 (IDG Books, ISBN 0-7645-3035-6) from Computer Manuals 0121 706 6000**

Also, bear in mind that, as before in this column, I will name names and quote figures, but you must only regard them as approximates. Do not base your entire business strategy on the figures I quote, because I don't know what your business requirements are. It often takes a couple of days' consultancy work to provide accurate figures for a given company. But as I hate reading evasive articles, I'll give ballpark figures which I think are reasonable for the average small-to-medium-sized enterprise (SME).

Buying a server

Buy it from a reputable company. I know they charge more but you usually get what you pay for, not only in terms of reliability but also in terms of compatibility (important with the kind of server OS you will be running) and support.

Good names here are Compaq, IBM, HP, Olivetti, Apricot, NetFrame etc. Note the "etc". Just because I haven't named a supplier, doesn't mean it produces poor products. On the other hand, don't assume just because a supplier can make a reasonable PC, it can also make a good server. Buy from a manufacturer with a good track record in making servers.

Questions and answers

I have received a few readers' letters during the past couple of months, so let's deal with these.

Q. *"Regarding the problem posed by Gareth Wade in your April column, which was about the problem of displaying times greater than 24 hours in Access. It does not have the [h]:nn:ss format that Excel uses for displaying hours greater than 24. As far as I am aware there is no format, as such, that will solve this problem. Access has other uses for square brackets.*

"I experienced a similar problem when totalling the times in a relay race that took place over two days. I realised the total time for a team could be greater than 24 hours and would then revert back to 00:00. I fixed it in a hurry by displaying just the minutes

Fig 1 There are several ways in which times can be manipulated

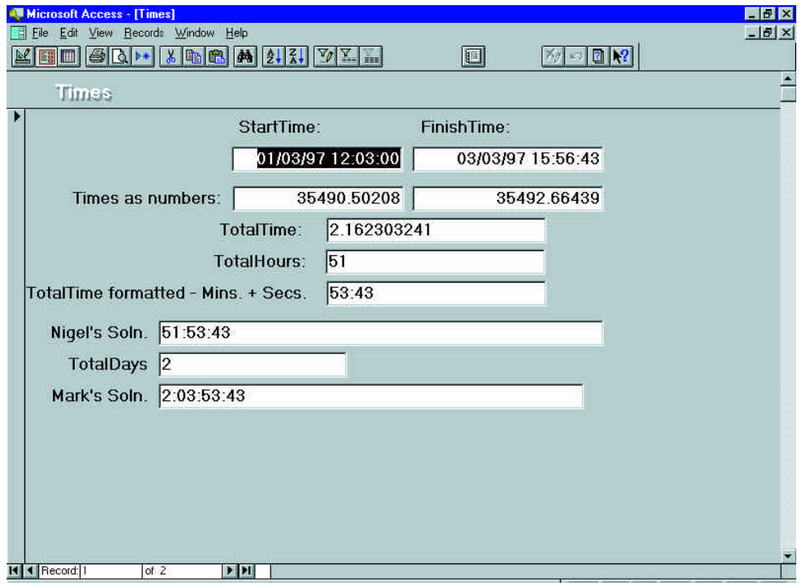
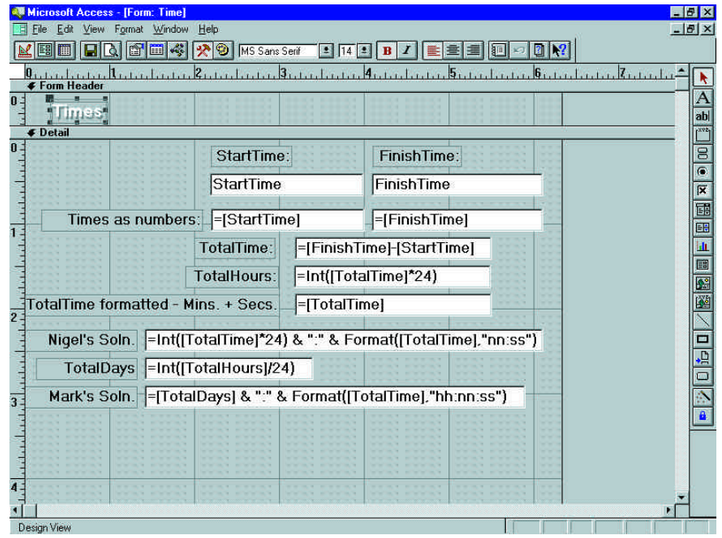


Fig 2 A design view of the same form *[see also Fig 1]*



*and seconds with nn:ss format and dealing with the hours separately. As I am sure you are aware, dates and times are stored as double precision numbers that represent the number of days after 30th Dec, 1899, so Int([TotalTime]*24) displays just the number of hours. The two parts can easily be combined into a single text box by making its control source*

```
=Int([TotalTime]*24) & ":" & Format([TotalTime], "nn:ss")
```

"While not actually being a formatting solution, this nevertheless seems to address the problem."

Nigel Collins

A. Nigel's solution is very neat. I have included a form which shows his solution, as well as some of the intermediate steps (Figs 1 & 2). The first two text boxes just show data from two fields in a record. These are the start date/time and the finish date/time of some mythical event. These text boxes are formatted as General Dates. The next two show the date/times formatted as numbers with fixed numbers of decimal places. As Nigel points out, the dates/times are really double-precision numbers that represent the number of days after 30th December 1899.

In Fig 2, you see that the syntax of the control source for these text boxes is of the type

`"=[StartTime]"` – *not* `"StartTime"`

Unless you use this syntax, Access will not allow you to choose a format like Fixed for the text box (although you can still type that format in by hand).

The third row on the form shows one text box which is calculating the difference between these two numerical values, while the fourth is calculating the total number of whole hours between the two dates.

The fifth shows the difference formatted as minutes and seconds. Formatting in this way effectively tells Access to ignore the hours and show only the minutes and seconds. (Note that this uses "nn:ss" and not "mm:ss" as you might expect.)

Nigel's solution is shown next and elegantly combines these two methods of showing the time difference. This led me to realise that we can also calculate the total number of days relatively simply and hence display the information in days, hours, minutes and seconds if so desired, as shown in the final two text boxes in Fig 2. This is clearly not *better* than Nigel's solution, it's just different.

Q. *"In your March Databases column, you mention the subject of storing hierarchical data in an SQL database.*

"The scheme you outline, which essentially uses a pointer to the next higher item in the hierarchy, is conceptually simple and very easy to understand. Sadly, as you noted, it is a pain to program with. You may be interested to know of a couple of articles written by Joe Celko. They were published in the weekly magazine Computing *(19th January 1995 and 26th January 1995 editions).*

"The concept he describes uses nested sets to represent hierarchical levels. Celko gives an example, using a simple organisation chart to demonstrate the tree structure. He also outlines a set of queries which allows you to do important things like find all the leaf nodes (those which have no further dependents) and identify the hierarchical chain of command from any individual, from the lowest to the top level.

"Please do follow this up and publish something in your column. Although relational databases are very good and very useful, there is nevertheless still a lot of data in the world which is hierarchical."

Richard Howells

A. Joe Celko is an American writer whose work on SQL I know and regard highly. Those interested in more efficient ways of storing hierarchical information would benefit from obtaining and reading these articles. Or send me some email, and if enough people are interested in the subject, I'll follow it up for you.

Q. *"I run a photographic model agency and we keep details of our models' characteristics, such as hair and eye colour, on our Access database. Sometimes we receive requests for "all girls with blonde hair". This is easy, as I can prepare a query for "blonde". However, I cannot seem to be able to request "all" hair colours. There appears to be no way of presenting a wild-card search as one of the input fields on the query through "[Input hair colour:]".*

"Can you think of any way of handling this criterion without resorting to non-Access code such as Basic?"

Mike Illes

A. I presume you're using parameter queries, in which case this syntax may be useful:

```
Like [Input hair colour:] & "*"
```

If you enter Blonde, it will find all Blondes. If you enter nothing, it will find all records. It also does a "fuzzy" search, so that you can mis-spell Blonde and still find the appropriate records.

PCW Contact

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column, at database@pcw.vnu.co.uk.

In the round

The subject of rounding in Access gets a discreet revival, and your contributions are sought for the most unusual RDBMS application. Mark Whitehorn briefs you on what's required.

This column has harboured several discussions about rounding functions in Access. I thought all had gone quiet until two intriguing emails arrived, one from Roger Moran and the other from Ray Hall. While some of us (myself included) find this topic fascinating, I am loath to devote much more space to it since it may be of limited interest to some people. So, I have included their emails in full as memo fields in the DBCMAY97.MDB file on the CD-ROM. This ensures that the information is available to those people who wish to look at it, but doesn't soak up bandwidth for those who aren't. See the form called "Rounding" if you are interested, and thanks to Roger and Ray for their contributions.

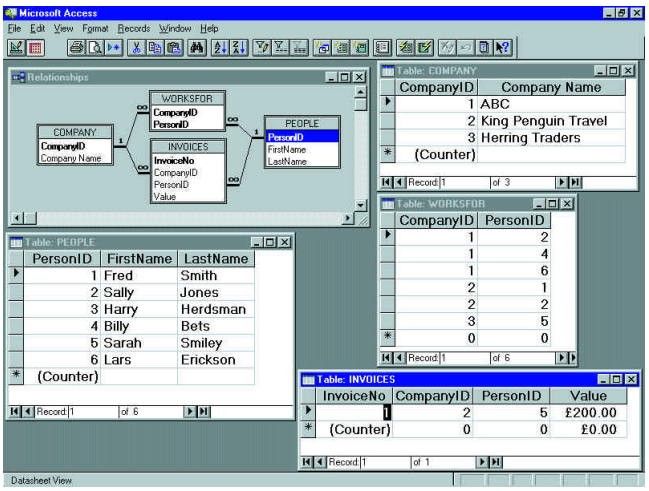


Fig 1 Taken from DBCMAY97.MDB and showing the tables from Greg Barstow's question

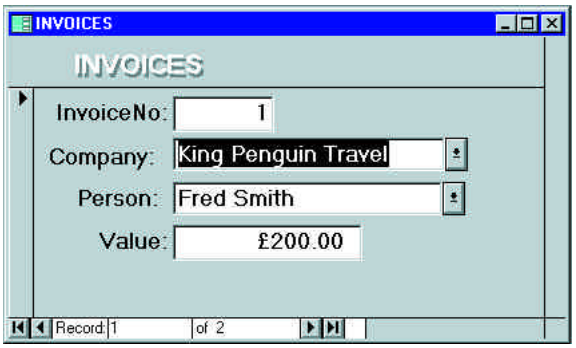


Fig 2 The two combo boxes on the form from DBCMAY97.MDB

Competition time!

From Andrew Leaman: *"I am interested in databases but find it difficult to think of useful applications. Could you possibly supply a list of typical applications, starting at the simplest and working up to the more complex?"*

Starting with the most basic database is easy: an address list, maybe a list for sending Christmas cards. As to the more complex uses, these are almost without number and range. Banking systems, air traffic control systems, process control systems in factories — all have at their heart some form of database. In fact, it is possible to argue that almost all computer applications are essentially databases; some of them just have rather odd front-ends.

Take a word processor, for example. It stores and manipulates data. You can query the database (with the search facility), generate different forms (normal view, outline view, etc.) and generate reports (print-out). You see? A document is really a database and a word processor is really a DataBase Management System.

I wouldn't want to take this argument too far, but what about a competition? We'll offer one of our much sought after book/record tokens for the most unusual example of an application developed with a recognised RDBMS. It doesn't have to be developed by you or your company, but if it happens to be so, that's fine. Just to start the ball rolling, I've heard of a really odd application, developed in Australia, which made the international news recently... But I'll leave it open for a reader to suggest that one since I'm sadly excluded from the list of potential prize-winners.

Sets and subsets

Greg Barstow writes: *"I do freelance for a range of companies. Each company has a number of people who can commission work from me for the company concerned. I need to generate invoices for each piece of work, and I want a form in Access which lets me pick the company from a combo*

p286 ➢

box (which I can do easily). Then I want the next combo box (which allows me to pick the person to whom the invoice should be sent) to show me only the people who work in that company."

This is a good generic question. Essentially it asks: "Given a long list of options which can be unequivocally sub-setted by a choice in another list, how do I show this elegantly on a form?" There are many applications. If you have different sales people who work on different product lines, or different aircraft which are serviced by different engineers, this is an area which may be of interest to you.

For one possible solution, see DBCMAY97.MDB. Fig 1 *(p285)* shows the tables involved and a small quantity of sample data. It also shows a tempting but incorrect set of relationships between the tables. Those who enjoy conundrums can work out why this particular set of relationships works but is non-optimal. The answer is on page 288 (the relationships in the MDB file on the CD-ROM are correct).

The form (Fig 2) has two combo boxes. The upper one is straightforward. It looks up values in the table COMPANY:

```
Select [CompanyID],[Company Name]
From [COMPANY];
```

and writes the value from COMPANY.CompanyID into INVOICES.CompanyID.

The lower combo box is more devious. It pulls data from a query called TheRightPeople rather than directly from PEOPLE. The purpose of this query is to find the people who work for the company which has been selected in the upper combo box.

The SQL for this query (Listing 1) is, like a great deal of SQL, impenetrable at first glance. Translated into English (more or less) it says:

- Find the value which is in the combo box above.
- Use that value to find the correct record in the COMPANY table.
- Then use that to find the people who work for the company. This has to be done via the table called WORKSFOR, since that is the table which stores the information about who works for which company.
- Finally, collect the relevant person's ID number and assemble their name neatly, attaching the first and last name together so that it looks tidy in the combo box.

If we run through that again, we can add in the relevant bits of the SQL statement:

Find the value which is in the combo box above.

```
[forms]![invoices]![companyID]
```

Use that value to find the correct record in the COMPANY table.

```
WHERE ((COMPANY.CompanyID=
[forms]![invoices]![companyID]))
```

Then use that to find the people who work for the company; this has to be done via the table called WORKSFOR, since that is the table which stores the information about who works for which company.

```
FROM PEOPLE
INNER JOIN (COMPANY INNER JOIN
WORKSFOR
ON COMPANY.CompanyID =
WORKSFOR.CompanyID)
ON PEOPLE.PersonID =
WORKSFOR.PersonID
```

Finally, collect the relevant person's ID number and assemble their name neatly, attaching the first and last name together so that it looks tidy in the combo box.

```
SELECT DISTINCTROW
COMPANY.CompanyID,
[FirstName]+" "+[Lastname] AS Name,
PEOPLE.PersonID
FROM PEOPLE
```

The nett result is that, when the second combo box is opened, the only names that appear belong to people who work for the company you have just chosen in the upper combo box. At least it *would*, except that it doesn't work automatically all the time because Access often buffers information so it doesn't automatically re-query the source for a control. The lower combo box has the query called TheRightPeople as its source. If this query has already returned an answer table, then simply selecting a different company in the upper combo box doesn't cause TheRightPeople to be re-run; hence the list of people may not be up to date when it appears in the lower combo box.

The answer is to force a re-query every time you use the lower combo box. This can be done in code or with a macro, and since I usually demonstrate everything in code, I thought for the sake of variety I'd use a macro (Fig 3).

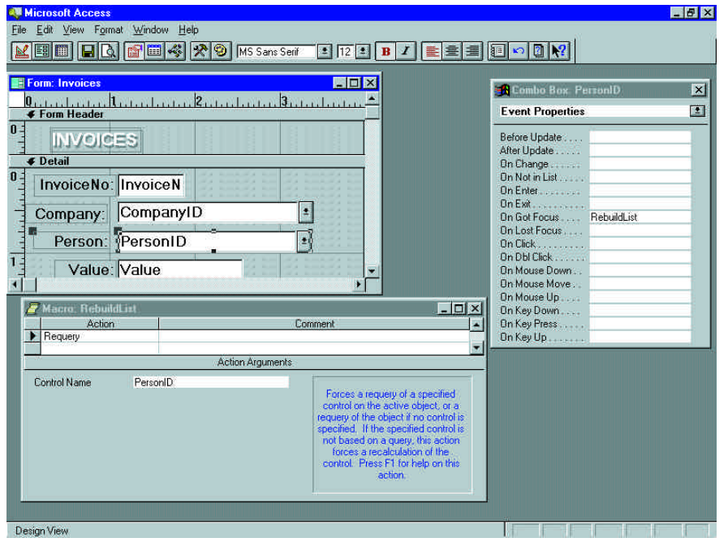


Fig 3 The extensive, and highly complex, macro used to force a re-query of the lower combo box on the form

David Ruffel emailed in a question which seems to have general application. He maintains a list of consultants who are experts in various areas — mathematics, computing, etc. There is a many to many relationship between the consultants and their subject areas, hence three tables are needed to model this relationship, while a fourth table contains information about times when the consultants are absent (Fig 4). Finding those consultants who can provide information about a given subject, say, Chemistry, presents no problem (Listing 2, p288).

However, David also maintains a table of the dates during which particular consultants are unavailable. What he needed was a query which found not only the consultants who were experts in a particular area, but also those who were available on a particular date.

To my twisted mind, the easiest way to solve this one is to use one query to find all of the Chemists who are unable to work. This can be accomplished with Listing 3 which looks pretty horrible if you aren't used to SQL, but is much more understandable in Access' GUI (Fig 4a). This produces an answer table which looks like Fig 5.

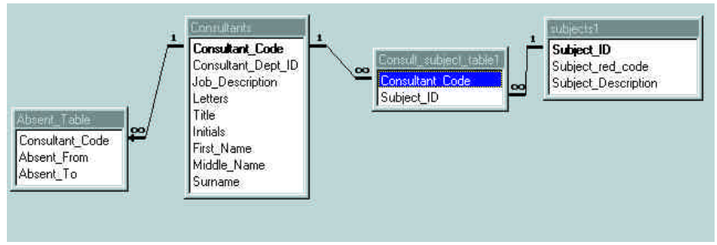


Fig 4 The tables used in David Ruffel's database

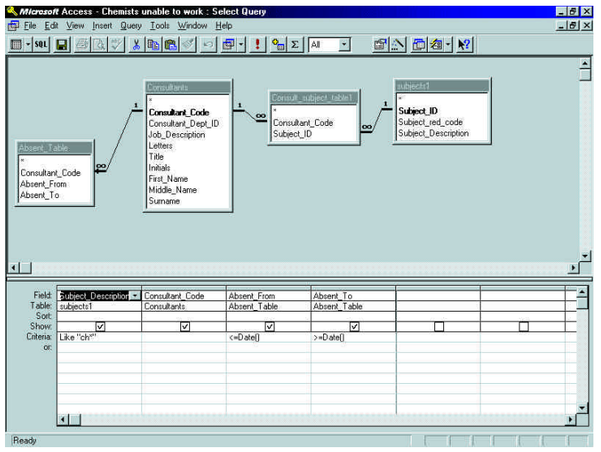


Fig 4a The GUI version of a rather impenetrable SQL statement

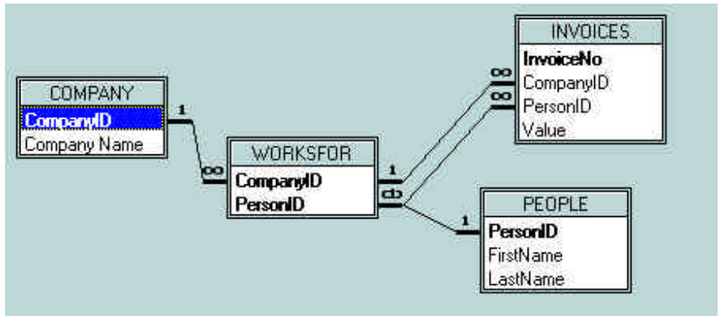


Fig 7 Showing a better set of relationships to use in DBCMAY97.MDB

Listing 1 SQL for TheRightPeople query

```
SELECT DISTINCTROW COMPANY.CompanyID,
[FirstName]+" "+[Lastname] AS Name, PEOPLE.PersonID
FROM PEOPLE
INNER JOIN (COMPANY INNER JOIN WORKSFOR
ON COMPANY.CompanyID = WORKSFOR.CompanyID)
ON PEOPLE.PersonID = WORKSFOR.PersonID
WHERE ((COMPANY.CompanyID=[forms]![invoices]![companyID]));
```

The ConsultantCode identifies the Chemist who can't work on the given date (in this case I am using the Date() function to return today's date). Then, a simpler query can use the information in this answer table to identify all of those who can work (Listing 4).

This solution may not be optimal and I haven't tested it extensively, but you might want to use something akin to it if you have a similar problem.

The sample file is on the CD-ROM as an Access 7 file called QP4.MDB.

The format of this file (Access 7) brings me to another email, from Dave Milor.

Versioning

"Is there any specific reason why you write your articles with reference to Access version 2 but are using the Windows 95 interface? I have heard that there are problems with version 7 regarding speed and possible bugs."

I use Access 2.0 whenever possible simply because Access maintains compatibility in only one direction. If I provide a solution in Access 2.0, anyone using that version or later can read it. However, if I supply an MDB file in the most recent version, Access 97, only those people with that version can use it. All of the machines that I now use are running either Windows 95 or NT 4, which is why the screenshots of Access 2 appear as they do. When run under 95 or NT, Access 2 "acquires" the look and feel of these particular systems.

With regard to bugs, Access 7 certainly has them; but then, so does every bit of software I have ever seen (including my own). I haven't come across any which would make the product unusable.

The speed issue is more complex. Given any RDBMS, speed considerations can be split into two areas. First, there is how fast the interface runs. This is non-trivial, since a slow interface makes development work painful and will upset end-users. Second, there is data-processing speed: essentially the speed with which queries run. This is very different, and also clearly non-trivial. This second measure of speed is the one which people like myself love to benchmark, but we shouldn't ignore the interface speed, even if it is more difficult to quantify.

So, what about speed in Access 7? The interface speed is worse than Access 2.0, but the data processing speed is a bit better with certain queries. Access 97 (the

p288 ➢

Listing 2 Finding Chemistry consultants

```
SELECT DISTINCTROW subjects1.Subject_Description, Consultants.Title, Consultants.Surname
FROM subjects1
INNER JOIN (Consultants INNER JOIN Consult_subject_table1
ON Consultants.Consultant_Code = Consult_subject_table1.Consultant_Code)
ON subjects1.Subject_ID = Consult_subject_table1.Subject_ID
WHERE (((subjects1.Subject_Description)="Chemistry"));
```

Listing 3 Finding the Chemists who are unable to work

```
SELECT subjects1.Subject_Description, Consultants.Consultant_Code, Absent_Table.Absent_From,
Absent_Table.Absent_To
FROM subjects1
INNER JOIN ((Consultants LEFT JOIN Absent_Table
ON Consultants.Consultant_Code = Absent_Table.Consultant_Code)
INNER JOIN Consult_subject_table1 ON Consultants.Consultant_Code = Consult_subject_table1.Consultant_Code)
ON subjects1.Subject_ID = Consult_subject_table1.Subject_ID
WHERE (((subjects1.Subject_Description) Like "ch*")
AND ((Absent_Table.Absent_From)<=Date())
AND ((Absent_Table.Absent_To)>=Date()));
```

Listing 4 Identifying the Chemists who can work

```
SELECT DISTINCTROW [Able Chemists].Consultant_Code, Consultants.First_Name, Consultants.Surname
FROM Consultants
INNER JOIN [Able Chemists]
ON Consultants.Consultant_Code = [Able Chemists].Consultant_Code
WHERE ((([Able Chemists].Consultant_Code) Not In (Select Consultant_Code from [Chemists unable to work])));
```

Fig 5 Answer table from Listing 3

Subject Area Description	ConsultantCode	Away From	Until
chemistry	5	Thursday, January 02, 1997	Thursday, June 19, 1997

next version on) definitely requires a better machine to run the interface (in other words, it is even slower). However, the data-processing is markedly faster, even given machines of the same spec.

Fig 6 Entry into the INVOICES table

InvoiceNo	CompanyID	PersonID	Value
1	1	3	£200.00
2	1	4	£0.00

A client-server future

In the February issue I asked if the column was too biased towards Access and, more generally, what did people want me to cover in future issues. The area which was overwhelmingly popular as a new topic was practical information regarding client-server databases.

This email from Tom Cliff was typical:

"I've been building databases for a couple of years and all of them have been on standalone PCs. Some of these have grown and now need to be migrated to a client-server system, because the volume of data has increased or they need to become multi-user. What I need is a good overview/ description of what is involved. How much work is it? How do I actually do it?"

So, we'll start next month having a look at the hardware you need, then move on to actually installing a back-end RDBMS, setting up a database on it, connecting to that database from clients, and so on.

Answer to conundrum

The relationships shown in Fig 1 ensure that the values inserted in INVOICES.PersonID are drawn from the available list of people (which is kept in table PEOPLE). They also ensure that the values inserted into INVOICES.CompanyID are drawn from the list of available companies. What these relationships *don't* forbid is an entry into the INVOICES table like Fig 6. This is bad, because person 3 doesn't work for company 1.

A much better set of relationships to use are those shown in Fig 7 *(p287)*. These ensure that the INVOICE table can only ever contain "meaningful" combinations of CompanyID and PersonID.

PCW Contact

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column, at database@pcw.vnu.co.uk

All buttoned up

Mark Whitehorn presents another slant on last month's buttons and captions query, which is far from sewn up. Plus, the complexities of data entry in Access, and CUSTOMER care.

In the February issue I published a letter from Glen Rowe. He wanted multiple buttons on a form, all with different captions. Whenever a button was pressed, a query would run which returned values based upon the caption on the button. For example, if you press the button labelled "Penguins" you see the records which relate to Penguins: pressing the "Fish" button yields information about Fish. I produced a solution by building a query which snatched the caption from the button which had just been pressed.

I also wrote the following: "The obvious solution at first is to try to pass the button's caption to the query as a parameter. As far as I know (and I stand to be corrected) this can't be done."

James Talbut replied: "Passing a parameter to a parameter query in Access can be done, but it's not very pleasant and means the you need to use DAO (Data Access Objects), which can be good or bad depending on the circumstances."

The solution that James then provides does, as he suggests, pass a parameter to a parameter query. What it doesn't do is pass the *caption* of the button as a parameter (which is still, as far as I know, impossible). In turn, this means his solution is somewhat more awkward than the one shown in the February issue because every button you add to the form must have its caption *and* its OnClick property altered. With the February solution, all you had to do was clone the button and alter the caption. However, James's solution to the initial problem is still well worth studying since it illustrates a very different way of solving the conundrum.

He continues: "Before I include all the bumf to show you how it's done, I'll just mention the implications.

"To actually access the resultant data you need to use DAO to step through a recordset. If you have a large amount of data this can be inefficient (OK, it's always inefficient, but it's more noticeable with a large dataset), particularly if you are intending to perform some secondary operation/selection on the data. However, the query itself is still run by the JET engine so the main data manipulation routines are still run as quickly as Access can do them.

"To control the parameters you need to use the Parameters collection of the QueryDef object for the query you are interested in. The most simple example I could come up with looks like this:

Table: "People"

ID	Name	Value
1	Fred	1
2	Jack	4
3	Fred	9
4	Harry	16
5	Anastasia	25
6	Bob	36
7	Martin	49
8	Stephen	64
9	Harry	81

(See Fig 1.)

```
Fig 1 Query, "Person":
SELECT DISTINCTROW People.ID, People.Name, People.Value
FROM People
WHERE (((People.Name)=[Person]));
```

```
Fig 2 Form Code, behind "People"
Private Function Button_Click(sName As String)
Dim qdfPeople As QueryDef
Dim rsPerson As Recordset
Dim sRowSource As String

 Set qdfPeople = CurrentDb.QueryDefs("Person")

  qdfPeople.PARAMETERS("Person") = sName
  Set rsPerson = qdfPeople.
OpenRecordset

  While Not rsPerson.EOF
    If Len(sRowSource) > 0 Then sRowSource = sRowSource & ";"
    sRowSource = sRowSource & rsPerson.Fields("Name") & " - " &
 rsPerson.Fields("Value")
 rsPerson.MoveNext
  Wend
  List10.RowSourceType = "Value List"
  List10.RowSource = sRowSource
End Function
```

p286 ➢

Form, “People”:
Six command buttons, each with their OnClick property set to:

```
=Button_Click(“Fred”)
```

with Fred replaced by the caption on that particular button. The buttons are labelled Fred, Bob, Harry, Martin, Anastasia and Stephen. At the bottom is an empty list box called List10.
(See Fig 2, page 285.)

“And that’s it. Hope it’s useful.”

Very interesting, James. I have constructed an example file (in Access 7.0) based on this example called TALBUT.MDB which is on the CD.

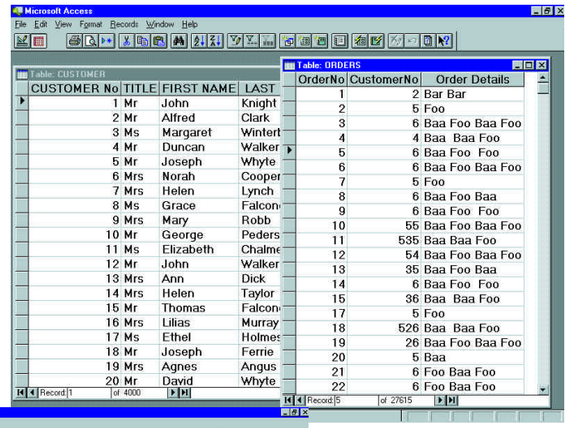
Next an email from Norway: “I have a problem that I’ve worked on for several months now, without finding any easy solution,” *writes Tore Saetre, of Bergen*. “I’ve developed and am maintaining a Microsoft Access database that keeps track of customers. The database contains two main tables: Customer and Sales. The problem is that I need to have a field in the Customers-table that shows how many orders the customer has in the Sales-table.

“My first idea was to make a query that counts orders in Sales for each customer. The query lists each customer and how many orders the customers has. I tried to move that value to the Customer-table with an update query that updates a NoOrders-field in Customer through a link to the first query. The problem is that the first query can’t be updated and the second query won’t update my Customers-table.”

I receive many questions like this one, questions which hinge upon the desire to store redundant data in tables. My initial reply to Tore was that, in general, storing derivable data is a bad idea because if the data changes (as you make more sales), the data in the CUSTOMER table goes out of date. It is normally preferable to use a query to calculate the information you need: whenever you want to see this data, you can run the query which shows you the customer details and the number of sales.

Tore was only partially convinced and replied: “But data in the sales table is only changed and added to once a month. The rest of the month the user browses through CUSTOMER’s 4,000 records and needs to see as much data on each customer as possible. Running a query for each customer is too time-consuming. I see why it can’t be done in a ordinary customer/sales database, but I still hope to find a solution to this problem.

Fig 3 (right) Small samples from both the CUSTOMER and ORDERS tables in the database “Tore”



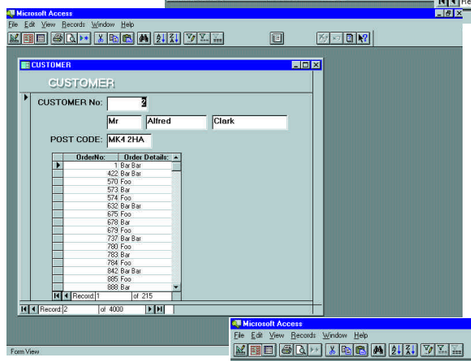
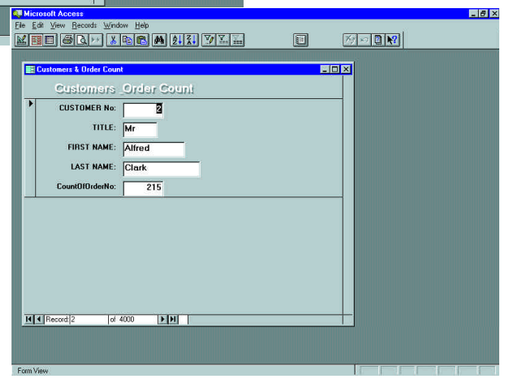


Fig 4 (left) The form called “Customers & Orders” shows customer details and order details

Fig 5 (below) The form called “Customers & Orders Count” shows one record for each customer



“For an experiment, do you have an example of how I can run a query to see customer details based on the customer currently displayed in a form? Do I need to run a code, macro or function and how do I get the query to filter everything but the record on-screen? Thanks for your help.”

Okay, we’ll solve this one both ways. There is a sample database on the cover CD (in Access 2.0) called TORE.MDB. This has two tables ,CUSTOMER and ORDERS. CUSTOMER contains simple details of 4,000 mythical people, ORDERS contains about 27,500 orders. Each customer has at least three orders to their name (or, in this case, to their CustomerNo) and some have considerably more. Fig 3 shows small samples from both tables.

The form called “Customers & Orders” (Fig 4) shows customer details and order details. The main form is based on CUSTOMER, the sub-form (which is called Sub1) is based on a query which simply performs a join on the two base tables. This form displays one record for each of the 4,000 customers.

The form called “Customers & Orders Count” (Fig 5) again shows one record for each customer. However, instead of listing the details about each order, it simply shows how many orders each customer has placed. This form is based upon the query called “Customer & Order Count”.

These two forms are variants which answer Tore’s request for an “example of how I can run a query to see customer details based on the customer currently displayed in a form”.

I ran this database on a 486/100 with 16Mb of RAM using Access 2.0. “Customers & Orders” opens almost instantaneously, and you can scroll through the records at the rate of about 400 per minute. “Customers & Orders Count” takes about 25 seconds to open, but once open you can scroll through the records at the rate of about 1,600 per minute.

To put this into simplistic terms, the first one does the calculations for each record as you ask to see it (which is adding about 0.1 seconds to the scroll time between each pair of records). The second one does all of the calculations for all of the records before showing any of them to you.

These forms are simply based on queries: no code or macros are required. As you can see, using the first form, it is possible to see customer and order details with essentially no speed hit at all for these numbers of records.

The query called “Customers & Orders Count” (the one which takes 25 seconds to run) can also be used, if required, to create the base table that Tore requested. I have included a Make-Table version of the same query in TORE.MDB. If you run this, it will create a base table called CUST which is just like CUSTOMERS except that it includes a field which shows how many orders each customer has placed.

In Tore’s case this could be run once a month, and then CUSTOMERS replaced by CUST; in that case, the 25-second speed hit disappears. The downside is that we are now reliant upon redundant data in the database. If anyone forgets to renew the tables at the end of the month, then all of the users of the database start to work with inaccurate data. You pays your money, you takes your choice — speed or security.

Quickies

■ Here’s a quick one from Gareth Wade: “When running a simple Report, I need to add all the time lengths and give a running

Fig 6

```
Private Sub LastName_AfterUpdate()

End Sub
```

Simply add a line so that it now reads;

```
Private Sub LastName_AfterUpdate()

    Me!LastName = StrConv(Me!
LastName, 3)

End Sub
```

Fig 7

```
Me!LastName = StrConv(Me!LastName, 3)
```

Fig 8

```
Private Sub LastName_AfterUpdate()
Dim Length As Integer

Me!LastName = StrConv(Me!LastName, 3)
If Left(Me!LastName, 3) = “Mac” Then
   Length = Len(Me!LastName)Length - 3)
   Me!LastName = StrConv(Me!LastName, 3)
   Me!LastName = “Mac”  Me!LastName
End If

If Left(Me!LastName, 2) = “Mc” Then
   Length = Len(Me!LastName)
   Me!LastName = Right(Me!LastName, Length - 2)
   Me!LastName = StrConv(Me!LastName, 3)
   Me!LastName = “Mc” + Me!LastNameEnd If

End Sub
```

p288 ➢

total. Easy enough; but when Access gets to 23:59, it reverts back to 00:00. What format do I use to carry on past that magic 24-hour mark ? I have tried all the manuals and help files but can't find the answer."

I don't know the answer to this one. Anyone else care to solve it?

■ And this one from Malcolm Rowley: "I find I am put off by the complexity required in Access to solve simple problems, e.g. data entry character formatting.

In Alpha I simply have to specify the word/character format I require, e.g. Upper, Lower, Word (first character upper case, the rest of the data left as is) in the field rules, and this is applied to all data entry that takes place and can be re-applied to all existing data. This is ideal for data entry of names and addresses; the only difficulty being the McCartneys (etc.) of this world. To have any simple form of formatting that stores the data in an access table in a particular format, do I really have to resort to code? Is there a simple way of accepting data entry in Access and storing it as first character upper case, the rest left as is?

For example: macdonalds becomes Macdonalds / macIntyre becomes MacIntyre / 27 park avenue becomes 27 Park Avenue. A simple solution would be appreciated."

The answer is that you have to use code, but only one line so it may just meet your requirement for simplicity! Suppose you have a form called Capital which displays a field called LastName in a text box called LastName. Switch to design, double-click on the textbox in question, select the Event properties, click on the one called "After Update", click on the ellipsis button which appears (three dots) and choose Code Builder. Between the two lines which appear (Fig 6) "Me" means the current form, "LastName" is the name of the textbox, and "StrConv" is a function (only available in Access 7.0 and above) which converts strings. The "three" tells the string conversion function to do the type of conversion that you asked for (capitalising the first letter of each word).

So, Fig 7 means "make the contents of the textbox called LastName equal to the same as it is now, but with all of the first letters of the words capitalised". This will convert: penguin penguinsson to Penguin Penguinsson / 23 the larches to 23 The Larches / mcdonald to Mcdonald and so on.

■ On a slightly different tack, I haven't used Alpha for well over a year, but I agree that

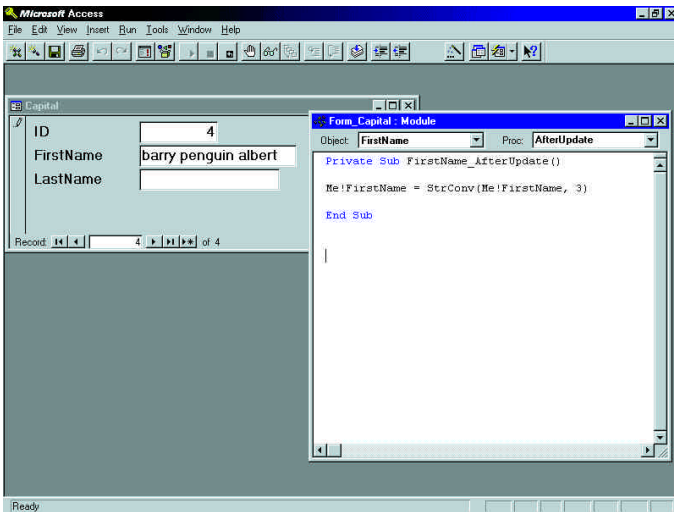
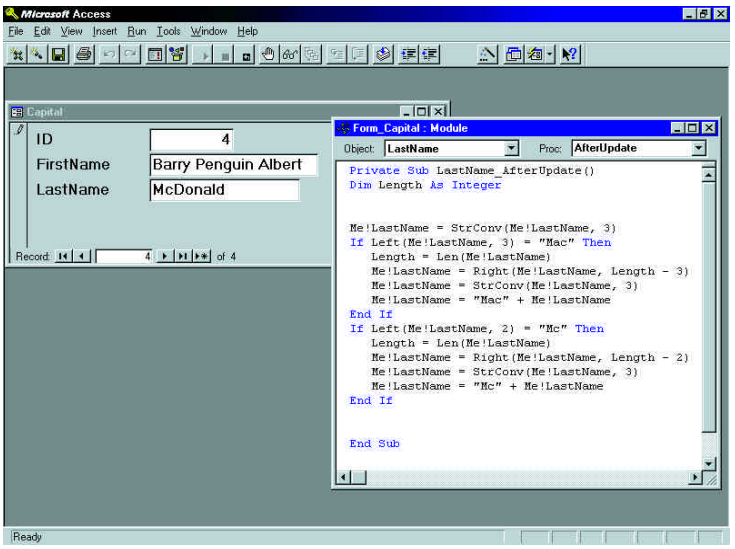


Fig 9 (left) This single line of code will capitalise all distinct words placed in the FirstName field *(see Fig 10)*

Fig 10 (below) This more complex code capitalises two of the more complex surname types in the LastName field



processes like this one are easier (and certainly more intuitive) in Alpha than in Access. In fact, to quote from a review I wrote at the time of its release: "…of these, the Field Rules are the most impressive, if only because Alpha copes with them better than any other RDBMS I have seen."

There is clearly a trade-off in design terms, which these two products exemplify. In both products, the designers identified a host of commonly needed functions and made them easily available from the interface. This approach has pros and cons. The good news is that commonly performed processes (like designing tables and building queries) are easy: the bad news is that if you happen to want a function that the designers didn't provide, you suddenly have to go to a more complex area, such as coding.

One difference between the two products is where the line was drawn. In this case, the Alpha designers decided to include capitalisation in the "easy" set and the Access designers didn't.

I have always found that, whatever product I use, I eventually reach a stage where I have to use code simply because it is impossible for the designers to put everything into the "easy" set. On a happier note, once you do start to use code, your horizons expand considerably. You can, for example, begin to deal with unusually capitalised names like those you mention. If you expand the code to that shown in Figs 9 and 10, this will change macdonalds to MacDonalds, and mctavish to McTavish. (Also see the Access 7.0 sample database called ROWLEY.MDB.)

I am not suggesting that this is perfect code. As usual in the sample code I give, there is no error trapping. In addition, there are some people who prefer the letter after Mac or Mc to be left in lower case. This code is simply provided as an example of what can be done.

PCW Contact

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column at database@pcw.vnu.co.uk