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

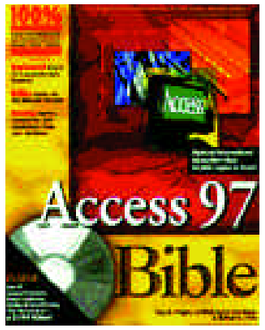
So, as you will have guessed, we have reached the part of the column where I try to get you to spend a small fortune on your server hardware. Please understand that as I try to part you from your hard-earned cash, I don't have shares in any of the hardware vendors who may be mentioned here. Rather, I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

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

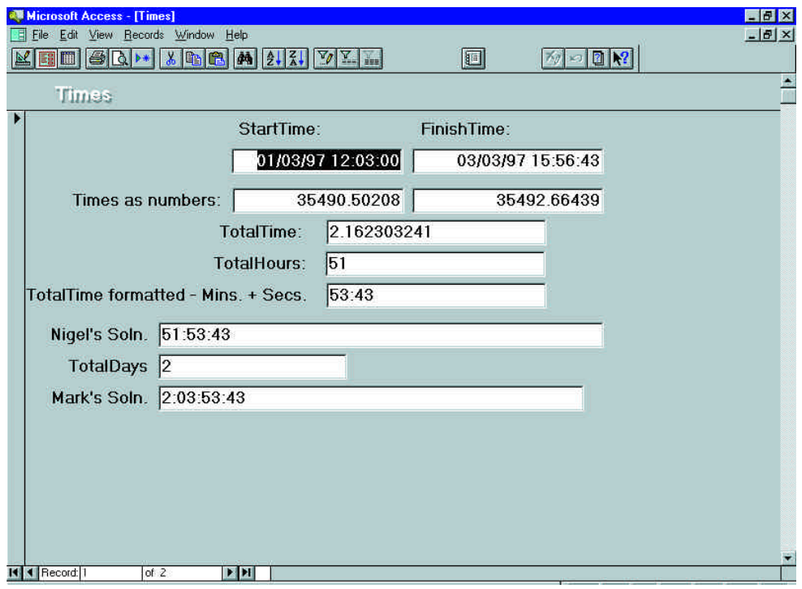
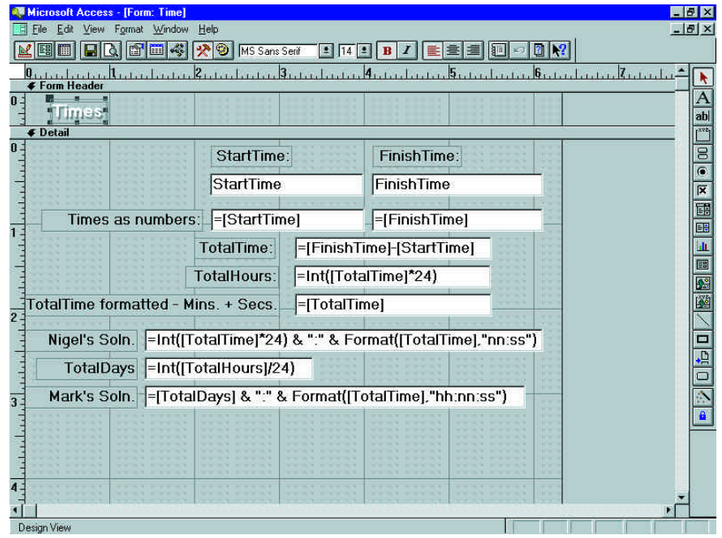


Fig 2 A design view of the same form *[see also Fig 1]*



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.