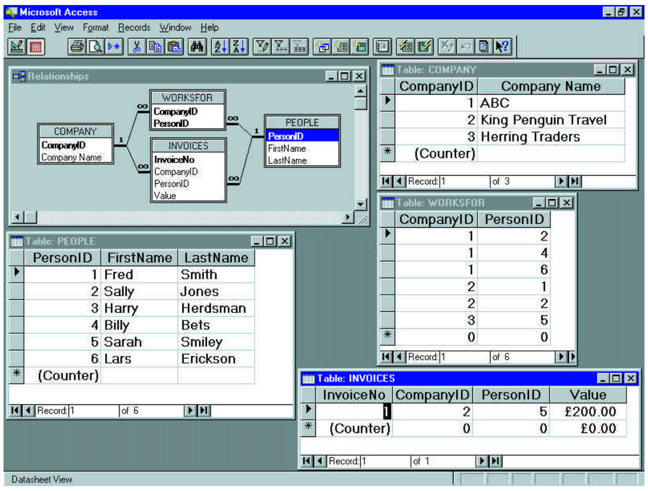
In the round

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.

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?"*



Microsoft Access — Relationships: COMPANY (CompanyID, Company Name); WORKSFOR (CompanyID, PersonID); INVOICES (InvoiceNo, CompanyID, PersonID, Value); PEOPLE (PersonID, FirstName, LastName)

Table: COMPANY

CompanyID	Company Name
1	ABC
2	King Penguin Travel
3	Herring Traders
(Counter)	

Record: 1 of 3

Table: WORKSFOR

CompanyID	PersonID
1	2
1	4
1	6
2	1
2	2
3	5
0	0

Record: 1 of 6

Table: PEOPLE

PersonID	FirstName	LastName
1	Fred	Smith
2	Sally	Jones
3	Harry	Herdsman
4	Billy	Bets
5	Sarah	Smiley
6	Lars	Erickson
(Counter)		

Record: 1 of 6

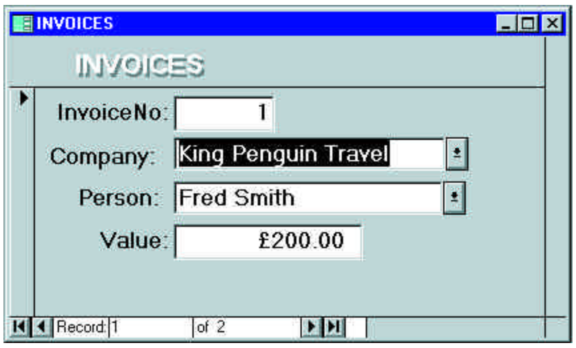
Table: INVOICES

InvoiceNo	CompanyID	PersonID	Value
1	2	5	£200.00
(Counter)	0	0	£0.00

Record: 1 of 1

Datasheet View

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



INVOICES

InvoiceNo:	1
Company:	King Penguin Travel
Person:	Fred Smith
Value:	£200.00

Record: 1 of 2

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

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:

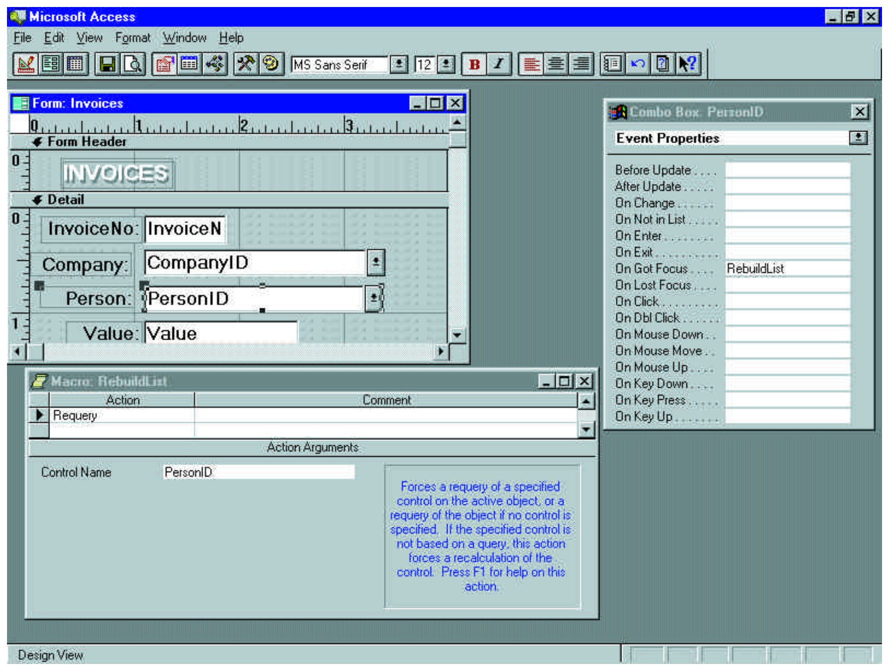
```
Select [CompanyID],[Company Name]
From [COMPANY];
```

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



Microsoft Access

File Edit View Format Window Help

Form: Invoices

Form Header

INVOICES

Detail

InvoiceNo:	InvoiceN
Company:	CompanyID
Person:	PersonID
Value:	Value

Combo Box: PersonID

Event Properties

Before Update	
After Update	
On Change	
On Not in List	
On Enter	
On Exit	
On Got Focus	RebuildList
On Lost Focus	
On Click	
On Dbl Click	
On Mouse Down	
On Mouse Move	
On Mouse Up	
On Key Down	
On Key Press	
On Key Up	

Macro: RebuildList

Action	Comment
Requery	

Action Arguments

Control Name PersonID

Forces a requery of a specified control on the active object, or a requery of the object if no control is specified. If the specified control is not based on a query, this action forces a recalculation of the control. Press F1 for help on this action.

Design View

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

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

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]));
```

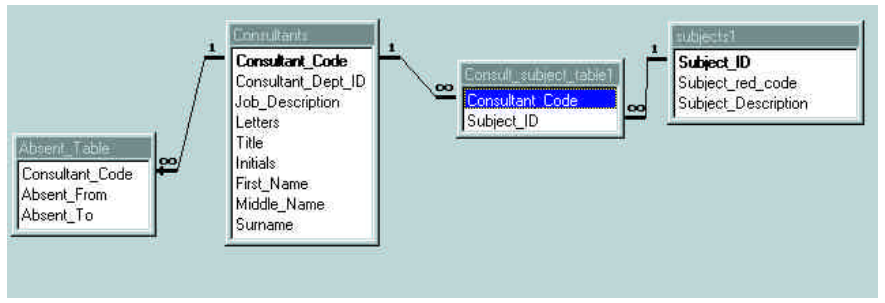
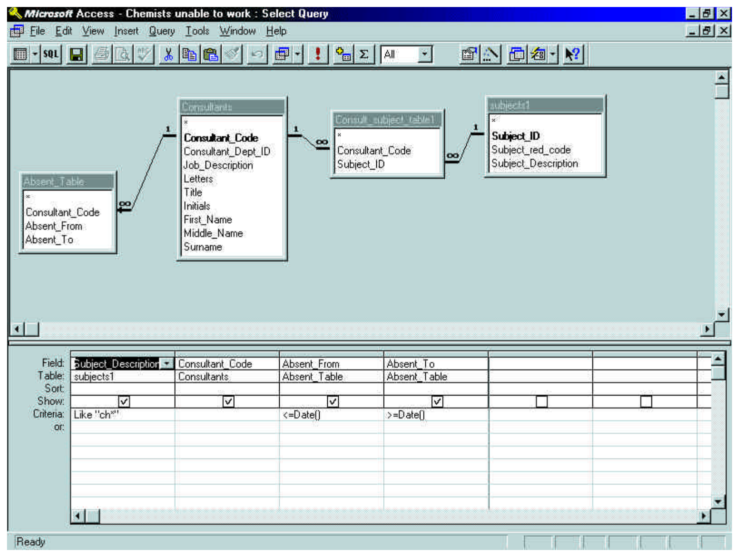


Fig 4 The tables used in David Ruffel's database



Microsoft Access - Chemists unable to work : Select Query

File Edit View Insert Query Tools Window Help

Field:	Subject_Description	Consultant_Code	Absent_From	Absent_To		
Table:	subjects1	Consultants	Absent_Table	Absent_Table		
Sort:						
Show:	✓	✓	✓	✓	☐	☐
Criteria:	Like "ch*"		<=Date()	>=Date()		
or:						

Ready

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

since I usually demonstrate everything in code, I thought for the sake of variety I'd use a macro (Fig 3).

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.

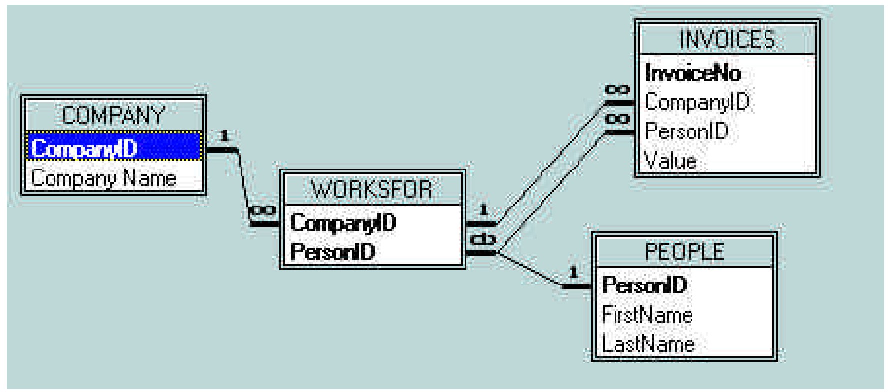


Fig 7 Showing a better set of relationships to use in DBCMAY97.MDB

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

Fig 6 Entry into the INVOICES table

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

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.

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