# Random pleasures

**Mark Whitehorn differentiates between random order and a lack of order for the sake of records.**

I found Linda Ratcliffe's <linda@ratcliffe15.freeserve.co.uk> question an interesting one: 'I want to randomly select and print about 20 records from an Access 97 table which contains around 800 records. I can find nothing in my Access 97 book or in the software help files.'

The problem is that databases tend to be rather pious about the order of the records in a table. On one hand, the relational model doesn't officially recognise the concept of 'order' in a table, so when you query a table, you are told to make no assumptions about the order in which the records are returned unless you specify an order, which is fair enough. However, a lack of order isn't the same as random order, so we can't just take the top 20 records from a normal query, because in practice we will get the same records over and over again. Ultimately, we get the worst of both worlds: no guarantee of order, but no real lack of order either!

Linda : Table

| ID | Foo | Baa |
|---|---|---|
| 1 | afds | j |
| 2 | afg | hjd |
| 3 | fg | gj |
| 4 | fg | gfhj |
| 5 | afg | gfh |
| 6 | fda | swtr |
| 7 | h | hse |
| 8 | sth | hjs |
| 9 | th | trhte |
| 10 | th | j |
| 11 | t | rsthj |
| 12 | afg | tdgj |
| 13 | adf | rsth |
| 14 | gad | jy |
| 15 | f | srth |
| 16 | dah | s |
| 17 | dsah | ytjs |

Record: 1 of 200

◀ FIG 1 A SIMPLE TABLE FROM WHICH TO SELECT RANDOM RECORDS – NOTE THAT IT CONTAINS 200 RECORDS

Given a table as shown in Fig 1, the most elegant Access solution is probably something like:

```
SELECT TOP 20 ID, Foo, Baa
FROM Linda
ORDER BY Rnd(ID);
```

(*Key:* ↙ *code string continues*)

This feeds the unique ID value into the random function which generates a random number for each record. The Top 20 bit then slices off the 20 records with the highest random number. You might be tempted by the slightly simpler:

```
SELECT TOP 20 ID, Foo, Baa
FROM Linda
ORDER BY Rnd();
```

Access Specific : Select Query

| ID | Foo | Baa |
|---|---|---|
| 133 | qer | reg |
| 69 | gfds | ds |
| 130 | jk | rg |
| 37 | gfds | hg |
| 95 | ru | ry |
| 7 | h | hse |
| 122 | yte | ujte |
| 139 | yj | tew |
| 101 | gwt | tr |
| 24 | f | fdsg |
| 140 | y | t |
| 103 | wy | try |
| 115 | rth | tty |
| 188 | sdf | u4m |
| 162 | we | fsgh |
| 82 | fdg | dgy |
| 135 | theyt | req |
| 148 | jtey | f |
| 92 | ut | w |
| 79 | gdf | rhs |
| (AutoNumber) | | |

Record: 1 of 20

◀ ...WE CAN THEN SELECT 20 RECORDS AT RANDOM

But that doesn't work because, given this SQL statement, Access takes the easier and presumably faster option of generating a single value using Rnd() and applying it to all records. The reason for feeding Rnd() with a 'seed' value from Linda.ID is to force it to generate a different random number for each record. As a matter of interest, if you use:

```
SELECT TOP 20 ID, Foo, Baa, Rnd(ID) AS Random
FROM Linda
ORDER BY Rnd(ID);
```

then it looks as if the sorting hasn't been correct because the values in the field called Random are not the highest values [Fig 2]. However, this is a reflection of the fact that Access is generating a value each time Rnd() is called, so a different value is being used for the ORDER BY clause.

One problem with this solution is that it's Access specific. It relies upon the fact that Access' Rnd() function will take a seed value; other RDBMSs may not do so in the same way. Additionally, the 'TOP' option isn't part of the SQL standard.

So I played around, looking for a more generally applicable solution. I didn't find a totally generic one, but the following may be of interest for RDBMSs which don't handle Rnd() in the same way or do not offer TOP 20.

## Solution 2

We can allocate a random number to each record and select those with the highest random numbers. We can add a column to the table which will hold such a number, and I have added such a column to the table Linda2.

We need to fill this column with random

▼ FIG 2 IF WE ASK TO SEE THE RANDOM NUMBER IT LOOKS AS IF THE SORT OPERATION HAS FAILED, BUT IT HASN'T STOPPED THE SQL STATEMENT FROM WORKING

Access Specific : Select Query

| ID | Foo | Baa | Random |
|---|---|---|---|
| 167 | ttw | sfg | 0.8404996 |
| 70 | gfd | fg | 0.707242 |
| 143 | ytej | hw | 0.8845419 |
| 47 | fg | fgh | 0.2075616 |
| 27 | sd | dszf | 0.1112442 |
| 103 | wy | try | 0.4694712 |
| 197 | hs | n | 0.1244656 |
| 58 | dfg | fgh | 0.1579564 |
| 66 | fds | hyrty | 0.4995615 |
| 29 | fdg | h | 0.8360133 |
| 60 | fdsg | fgh | 0.6803569 |
| 110 | gte | req | 0.7737299 |
| 124 | uk | tr | 0.0518994 |
| 5 | afg | gfh | 0.230025 |
| 150 | teyj | sg | 0.5952455 |
| 93 | yue | ywet | 0.9882925 |
| 3 | fg | gj | 0.7805259 |
| 14 | gad | jy | 0.1772022 |
| 192 | sf | rwr | 0.6045440 |
| 125 | utilo | t | 0.6579066 |
| (AutoNumber) | | | |

numbers, so I initially thought about doing this in the approved set operation way using an update query:

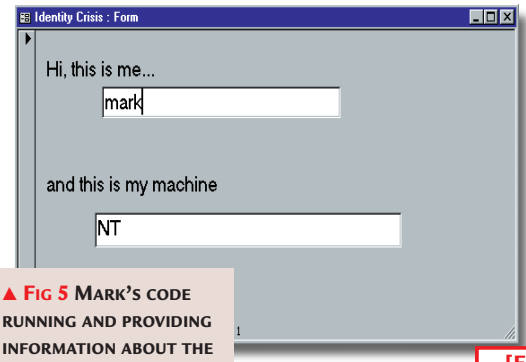
```
UPDATE Linda2 SET ↙
Linda2.RandNo = Rnd();
```

This is a great idea, but it doesn't work because Access generates one random number and updates all the records with that number.

So I wrote some code that loops through the table, writing a different random number into each record.

```
Do Until Linda2.EOF
  Linda2.Edit
  Linda2![RandNo] = Rnd
  Linda2.Update
  Linda2.MoveNext
Loop
```

That does the job, adding a random number between 0 and 1 into the field.




▲ FIG 5 MARK'S CODE RUNNING AND PROVIDING INFORMATION ABOUT THE USER AND THE MACHINE

Then we need a query that extracts about 20 of the largest random numbers (or we could use the 20 smallest, it doesn't matter). This is created in SQL like this:

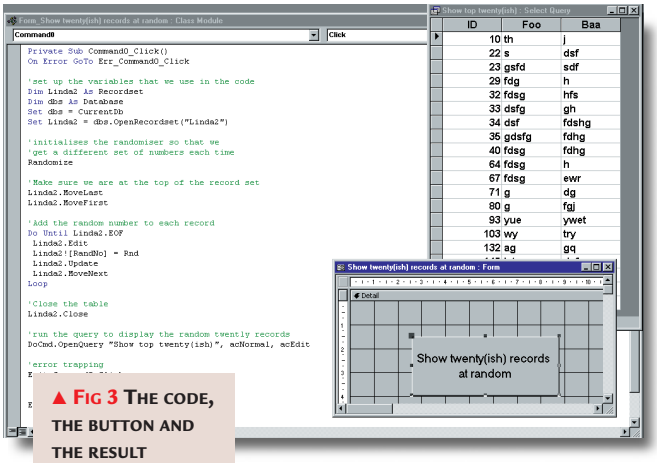
```
SELECT ID, Foo, Baa
FROM Linda2
WHERE (((RandNo)>0.9));
```

The value 0.9 is based upon the fact I have 200 records in the sample table, the random number is between 0 and 1, so >0.9 should give me about 20. (Please don't get into discussions about whether this should be >=0.9, it is only ever going to give approximately 20 records anyway!)

This particular statement does not display the random number in the answer table because that number has served its job by this point, but you can always add it.

Each time we want a random set, we generate a new set of random numbers in the table and run the query that pulls out about 20. We can automate the process using code and tie it to a button on a form – as in the sample database [Fig 3].

There are further solutions to this problem. We could, for example, write code to jump randomly in the table, picking up a record at each jump. Twenty jumps will give us 20 records (checking we don't hit the same record multiple times). This solution is better because we don't have to add a field to the table, but worse because the code is more complex.




▲ FIG 3 THE CODE, THE BUTTON AND THE RESULT

However, all the solutions I can think of either make use of Access-specific features, or don't return exactly 20 records (I know Linda didn't ask for exactly 20, but this additional condition makes the problem more interesting). You may be smarter than me, so I throw it open to the readers. But we are looking for totally generic solutions that use only set-based operations.

He sent in a 'Computer-name and User-name code (found on the Office 2000 installation "Excel sample file" with some minor adjustments by me)'. This is a VB code [Fig 4] that returns the Windows User Name and the Computer Name. This seems to work fine and the result is shown in Fig 5.

Mark is aware that every network card has a unique ID number and was wondering how he could get this number from within the Visual Basic for Applications environment. He would like to compare it with those numbers stored in a secure table, to tighten the security of his databases. Write in with your ideas.

[FIG 4]

```
Option Compare Database
OptionExplicit
Private Declare Function GetUserName Lib "advapi32.dll" Alias ↙
"GetUserNameA" (ByVal lpBuffer As String, nSize As Long) As Long
Private Declare Function GetComputerName Lib "kernel32" Alias ↙
"GetComputerNameA" ByVal lpBuffer As String, nSize As Long) As Long
'Get User name and get computer name functions.
Function Get_Computer_Name()
    Dim Comp_Name_B As String * 255
    Dim Comp_Name As String
    GetComputerName Comp_Name_B, Len(Comp_Name_B)
    Comp_Name = Left(Comp_Name_B, InStr(Comp_Name_B, Chr(0)) - 1)
    Get_Computer_Name = Comp_Name
End Function
Function Get_User_Name()
    Dim lpBuff As String * 25
    Dim ret As Long, UserName As String
    ret = GetUserName(lpBuff, 25)
    UserName = Left(lpBuff, InStr(lpBuff, Chr(0)) - 1)
    Get_User_Name = UserName
End Function
```

*(Key: ↙ code string continues)*

**Identity crisis**

Mark Boreham <mark.boreham@lucasvarity.com> is a student who is currently doing a degree specialising in databases.

**PCW CONTACTS**

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