Set practice

Mark Whitehorn on set operations and record ordering in Access.

From John Norton, janorton@crcs.demon.co.uk, comes a question: 'I want to know how I can print lists from a table or query in Access (presented in a form if necessary) which will number each record. Ten years ago, when I was using dBase III, I could do this simply by specifying 'recno()' after the print command. In nearly ten years of using Access I still haven't figured out how to do it!'

This raises some interesting points and concepts that touch upon Access and the relational model underlying it. Relational databases are designed to perform operations on 'sets' of records. Implicit in set operations is the idea that the records in the set are not ordered. There is no 'first' nor 'last' record and neither does the concept of a 'next' record mean something. This sounds weird at first but it does have major advantages.

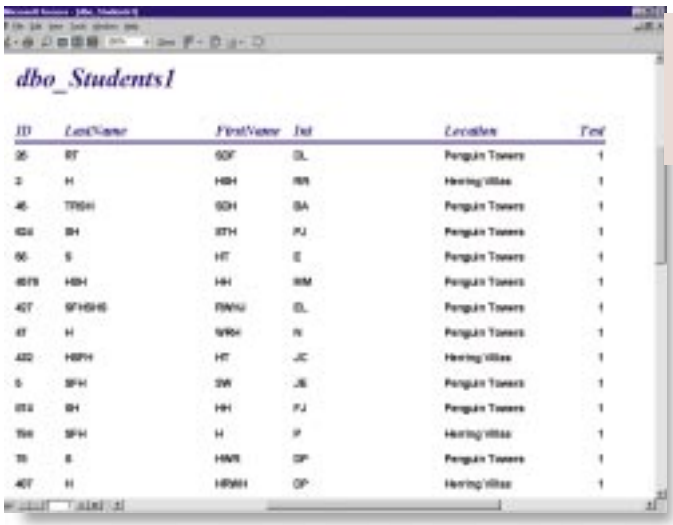
When you want to scan, say, five million records for those that relate to a particular city, you don't have to write:

```
Open Table 'Cities'
Repeat
      Select record if City
 = 'Hereford'
Skip to next record
Until end of records
Close Table 'Cities'
```

You just write:

```
Select * from Cities where
 City = "Hereford"
```

It is implicit in this one statement that all of the records in the set will be examined. Access is a much closer match to the relational model than dBASE ever was so Access shows very little interest in the concept of 'order' in record sets. In fact, you may already have noticed this when you run queries. Unless you explicitly tell Access how you want the records in the answer table to be ordered, they can appear in what looks like random order.



◂ FIG 1 A BASIC REPORT, WITH NO TRICKS UP ITS SLEEVE

Once you get used to the idea that relational databases don't care about the order and position of records, more aspects of their behaviour make sense. However, there is also no denying that humans tend to like the concept of 'order' and we often like the idea that the position of a record has some value or importance. Hence John's email.

John is a bird watcher and he wants his sightings to be numbered. This is perfectly fair from a human perspective but has no meaning from a relational point of view. *OK, Mark, so what you are telling us is that it can't be done, right?* Wrong.

Access shows little interest in the concept of 'order' in record sets

Despite this, we can do what John wants but we will have to do it with the non-relational part of Access.

What? You have just stated that Access is a relational database management system!…

So it is, which brings us to another interesting point.

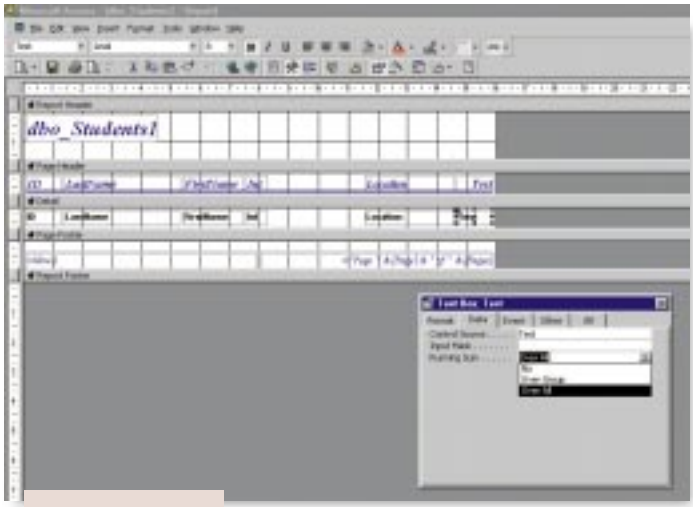
The relational model is all about the rigour that's required when we store data, modify it and query it. That same relational model has nothing to say, for instance, about how we print out the data — it's too trivial a process to concern the relational model. So, in fact, an RDBMS doesn't have to contain, say, a reporting tool. True, reporting is very useful and that is why a reporting section has been bundled with Access but if it were removed, that removal wouldn't stop Access being relational. There is no reason why the reporting tool cannot concern itself with numbering and ordering records, and so it does.

Instead of birds, the sample table contains student records. These are actually real records from a real system where I needed to number the records in the way John requires. Only the data has been changed to 'protect the innocent'.

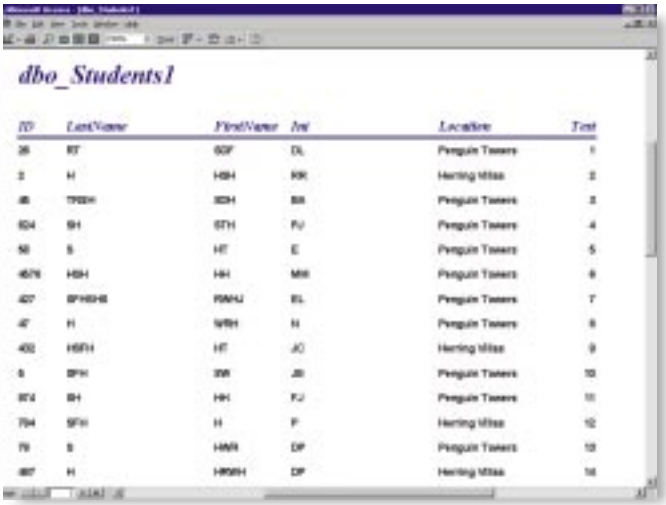
The student data consists of a table called 'dbo_Students' with the fields as follows: I have created a query called 'dbo_students1' (see Fig 1) which contains exactly the same fields and simply adds a field (imaginatively called 'Test') and ensures that in every record

FIG 2

ID	LastName	FirstName	Int	Location	Test
2	H	HSH	RR	Herring Villas	1
245	S	HT	TM	Herring Villas	1
2457	SH	SGT	L	Penguin Towers	1
26	RT	SDF	DL	Penguin Towers	1
3	SH	HT	LW	Penguin Towers	1
34	HS	TR	LA	Herring Villas	1



▲ FIG 3 USING THE SYSTEM TO NUMBER THE RECORDS WITHIN GROUPS



dbo_Students1

ID	LastName	FirstName	Int	Location	Test

▲ FIG 4 NUMBERED RECORDS ON THE REPORT

the value 1 is assigned to that field.

This field could have been added to the original table and the numbering would still work. However, putting all those identical number 1s in the original table would have meant storing redundant data, a process which would have offended the relational model, and we mustn't upset the model otherwise it sulks.

The SQL for this query is trivial:

```
SELECT dbo_Students.*, 1 AS↙
 Test
FROM dbo_Students;
```

(Key: ↙ code string continues)

and produces Fig 2. Now we can generate a report based on this query which lists all of the fields. This report is in the sample database STUDENTS .MDB on our cover-mounted CD as dbo_students1.

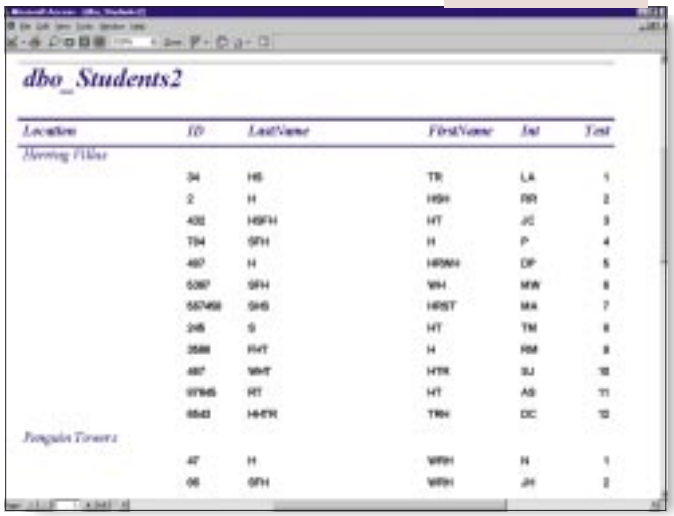
As you can see [Fig 1], the Test field is full of ones. Now flip into design mode, highlight the field, pull up its properties and select the Data tab. You will find that there is a property called Running Sum (which sounds for all the world like a Native American name!). Set this to Over All [Fig 3] and re-run the report. You should find that the test field is now numbering the records on the report [Fig 4]. This report is saved in the test database as dbo_students2.

At the risk of stressing the point too heavily, I know that we could use a counter field to number these records and get a numbered list like this. We could even order the records so that the records appeared in numbered order. However, such a system would only work if every record were included in the report. The system shown here will work, no matter what query is used to extract whatever subset of required records.

This facility has several other uses, most of which will be left as an exercise for the reader — in other words, have a play and see what happens. For instance, what happens if the value in the test field is not always a 1?

One particular use is worth spelling out because it is so handy. If you create a 'group by' report and set Running Sum to be Over Group, then the records within the group are numbered [Fig 5]. (This resulting report is saved in the test database on our CD as dbo_students3). So, Access *can* emulate the features that dBASE had ten years ago — you just have to dig a little deeper!

▼ FIG 5 USING THE SYSTEM TO NUMBER THE RECORDS WITHIN GROUPS



dbo_Students2

Location	ID	LastName	FirstName	Int	Test
Herring Villas					
	34	HS	TR	LA	1
	2	H	HSH	RR	2
	432	HSFH	HT	JC	3
	784	SFH	H	P	4
	467	H	HRWH	DF	5
	5367	SFH	WH	MW	6
	567468	SHS	HRST	MA	7
	245	S	HT	TM	8
	3588	FHT	H	RM	9
	487	WHT	HTR	SJ	10
	97845	RT	HT	AS	11
	8543	HSFH	TRH	DC	12
Penguin Towers					
	47	H	WRH	H	1
	66	SFH	WRH	JH	2

Also on our cover CD

I recently received an email about a form I demonstrated a while back which had been sent in by a reader and showed how colours could be used on a form to flag different levels of importance.

☛ You can use a combo box to choose a level of importance for a contact (Extinct, Cold, Hot, etc.) and a different colour is displayed on the form next to that record. So, in case anyone else is looking for it, I have included it again on our cover CD as COLOUR.MDB.

☛ In the March column, under *Hot Dates*, reference was made to an inclusion of set-based solutions on our cover CD. Unfortunately this was overlooked but you will find it on this month's CD in a special late-entry section.

PCW CONTACTS

Mark Whitehorn welcomes your feedback on the Databases column. Contact him via the PCW editorial office (address, p14) or email database@pcw.co.uk