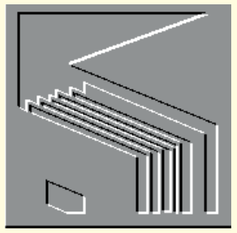


Meter **mania**

The meter readings problem has sent Mark Whitehorn's postbag level off the dial and offered up an alternative view.

The meter reading problem which first appeared in the March issue, continues to fascinate people (myself included). One reader is, however, convinced that I have missed the point. He feels that the real problem lies not in the SQL required to extract the data, but way back in the structuring of the original table I supplied. He is convinced that it isn't normalised to third normal form, and that the entire problem is easier to solve if the table is normalised to this level.

The original table structure I suggested for the meter problem…

Meter No	Date	Reading
1	18/05/91	20
1	11/11/91	91
1	12/04/92	175
1	21/05/92	214
1	01/07/92	280
1	21/11/92	270
1	12/12/92	290
1	01/04/93	824
2	18/05/91	619
2	17/09/91	712
2	15/03/92	814
2	21/05/92	918
2	17/09/92	1028
3	18/05/91	20612
3	11/11/91	21112
3	15/03/92	21143
3	21/05/92	21228
3	17/09/92	21456
3	21/03/93	22343

Challenging meter reader

The following is paraphrased from several email exchanges on the subject.

"The main problem stems from the fact that you have failed to identify that each meter reading is in itself an item of interest and should therefore have an individual unique identifier. It is often true that combining data fields in order to arrive at a unique index means that the data design is potentially incomplete. It can be useful to retrieve items in this way as it may speed up searching, but it is often more desirable to have a separate unique item key.

Reading No	Meter	Date	Reading	Previous Reading
1	1	18/05/91	20	20
2	2	18/05/91	619	619
3	3	18/05/91	20612	20612
4	2	17/09/91	712	619
5	1	11/11/91	91	20
6	3	11/11/91	21112	20612
7	2	15/03/92	814	712
8	3	15/03/92	21143	21112
9	1	21/05/92	214	175
10	2	21/05/92	918	814
11	3	21/05/92	21228	21143
12	3	17/09/92	21456	21228
13	2	17/09/92	1028	918
14	3	21/03/93	22343	21456
15	1	12/04/92	175	91
16	1	01/07/92	280	214
17	1	21/11/92	270	280
18	1	12/12/92	290	270
19	1	01/04/93	324	290

…and the table suggested by a reader. Which one is normalised? Which one will be faster to query?

"Thus, by adding (as a counter type) a ReadingID field at the front of your table and then the rather obvious PreviousReading at the end, we end up with a five-field table to deal with using ReadingID as the primary key.

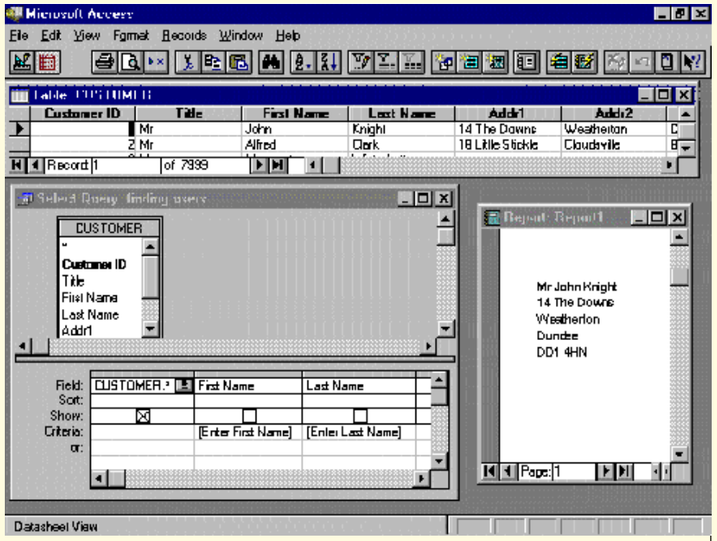
"PreviousReading contains the ID of the previous reading and, to make the initial conditions easier, can point to itself if there are no previous readings."

At this point, the discussion moved to one about third normal form:

"Nowhere does it state (in Codd's rules) that you can simply add together other fields to make up your unique identifier. The use of a single unique key is indeed needed in order to satisfy third normal form.

"This sort of 'I can add things together to make it fit the rules' approach causes more problems that any other aspect of database design. Please go away and re-read Codd. You will see that the only way the data can be produced in third normal form is by the addition of the field I suggest, not by combining two other fields to make it work.

"The solution I proposed in fact meets third normal form, in the same way as yours does. You may just argue that this extra field is redundant as it reproduces the information you already have, but that is to miss the basic advantages and simplification that it gives. You cannot claim that it breaks third normal form. (If you consider that each unique ID only relates to one combination of Meter/Date and vice-versa, you will see this.)



A parameter is easy to set up. Enclosing a criteria entry in [square brackets] means that, when the query runs, it will ask you for the relevant data (shown on the right of the screen)

"I repeat. If you have to add together two fields to make up a (primary) key, it is almost certain that you have missed the need for a unique identifier. Such was your case. Each record as taken by the meter-reading man stands alone as a separate item and should be identified as such. It may be convenient to refer to it by other fields, but that is not the point."

It is clear that we disagree fundamentally about the way in which this table should be structured in order to comply with the relational model. Inevitably, I think that I am right, but that proves nothing in absolute terms — have you ever met any database people who *didn't* think they were in the right?

More interesting than who is right or wrong is the fact that the two solutions provide an interesting compare and contrast exercise:

1. Which one do you think is flawed in terms of the relation model, and why?
2. To what update and delete anomalies does the flawed one lead?
3. Which one will be faster when queried?
4. What are the implications of using each table in a real database?

Singled out

"Can you tell me how to get access to print one label instead of a complete list?"
ENSC1EM1@tay.ac.uk

The trick is to base your report on a query which lists only the single label that you want to print. In fact, if you want to be flash, you can base the report on a parameter query. A parameter query is one which asks you for the condition (for instance, the name of the person for whom the label is destined) before it runs.

So, when you run the report, it will run the parameter query; the parameter query will ask you for the name, and the report will print the label!

Sounds complicated, but it should be simple to use.

Table 1

EmployeeNo	FirstName	LastName	DateofBirth	DateEmployed
1	Bilda	Groves	12/04/56	1/5/89
2	John	Greeves	21/03/67	1/1/90
3	Sally	Smith	1/05/67	1/4/92

Table 2

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
1	1	Simpson	Sofa	Harison	£ 235.67
2	1	Johnson	Chair	Harrison	£ 453.78
3	2	Smith	Stool	Ford	£ 82.78
4	2	Jones	Suite	Harisonn	£3421.00
5	3	Smith	Sofa	Harrison	£ 235.67
6	1	Simpson	Sofa	Harrison	£ 235.67
7	1	Jones	Bed	Ford	£ 453.00

Table 3

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
3	2	Smith	Stool	Ford	£ 82.78
5	3	Smith	Sofa	Harrison	£ 235.67
213	3	Williams	Suite	Harisonn	£3421.00
216	2	McGreggor	Bed	Ford	£ 453.00
217	1	Williams	Sofa	Harrison	£ 235.67
218	3	Aitken	Sofa	Harison	£ 235.67
225	2	Aitken	Chair	Harrison	£ 453.78

Table 4

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
1	1	Simpson	Sofa	Harison	£ 235.67
6	1	Simpson	Sofa	Harrison	£ 235.67

● *Please email me, or write in and let me know what* you *think.*

Is it a bird? Is it a bug?

The question of speed in the meter problem is also of some interest. The different solutions suggested clearly have different speed implications and different update potentials. I started (as promised) to have a look at this, but I have been unable to come to a final conclusion for the intriguing reason that some of the tests are hanging Access. Is it a bird, is it a bug, is it my PC? Microsoft and I are, as we say in the trade, engaging in meaningful discussions. I'll keep you posted.

And so, to SQL

It was clear from the meter problem that many people wanted the subject of SQL covered for its own sake, so here we go. But first, a little background.

Data manipulation is a really important part of the relational model. After all, there is little point in storing data correctly, safely and securely, if that is all you ever do with it. Stored data has no value if you cannot question it and extract the data in some way for humans to examine. SQL is the language typically used for data manipulation and this, like most languages, has a set of commands. These commands in their turn are based on a set of fundamental operators.

In order to get the best from SQL, it is worth having at least a passing acquaintance with these operators. They are rarely used directly, but in my experience they keep cropping up in conversations about SQL. Knowing nothing about them can leave you at a social disadvantage. Trust me, it's worth knowing.

The only qualifier I need to apply is that the definitions I give are not complete to the relational model. For example, I say below that: "In order for tables to be 'Union Compatible', they must have the same number of fields, and the fields must be of the same data type and size." In fact, to be truly Union compatible, the fields must also draw their values from the same domains.

However, this level of pedantry is counter-productive, since most RDBMSs do not even support domains. So, as in the past, I will sacrifice exact mapping on to the relational model for practicality/readability.

The relational operators

Most of us are familiar with the standard algebraic operators (+,-,* and /) which signify addition, subtraction, multiplication and division. We use these operators to manipulate numerical values, or variables which represent values, almost without thinking. Thus, if we know that

```
A=5, B=6 and C=10
```

and that

```
D=A+(B*(C/A))
```

we can calculate that

```
D = 5+(6*(10/5))= 5+(6*2) = 5+12 = 17
```

In a database, we store the data in tables (also known as relations) and the relational model provides a set of operators (known, therefore, as relational operators) with which we can manipulate tables (that is to say, relations). The discerning, sensitive reader will have noticed that I am showing a slight tendency to slip into "database-speak" at this point. I do so only because, without using the terms, expressions such as "Relational Operators" do not make much sense.

However, it is worth noting before we begin that tables are relations, and relations are sets. Four of the relational operators (Union, Difference, Intersection and Product) allow us to perform traditional set operations.

In order to demonstrate these operators we need a sample table or three *(see Tables 1, 2 and 3*).

If I enter "John" in answer to the first question, and "Knight" in answer to the second, then the parameter query will find his one record out of 7999. The report is based on the query, so it prints out only a single address

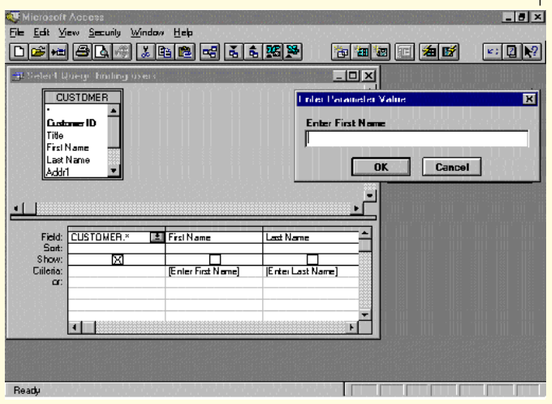


Table 5

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
1	1	Simpson	Sofa	Harison	£ 235.67
2	1	Johnson	Chair	Harrison	£ 453.78
3	2	Smith	Stool	Ford	£ 82.78
4	2	Jones	Suite	Harisonn	£3421.00
5	3	Smith	Sofa	Harrison	£ 235.67
6	1	Simpson	Sofa	Harrison	£ 235.67
7	1	Jones	Bed	Ford	£ 453.00
213	5	Williams	Suite	Harisonn	£3421.00
216	2	McGreggor	Bed	Ford	£ 453.00
217	1	Williams	Sofa	Harrison	£ 235.67
218	4	Aitken	Sofa	Harison	£ 235.67
225	4	Aitken	Chair	Harrison	£ 453.78

Table 6

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
3	2	Smith	Stool	Ford	£ 82.78
5	3	Smith	Sofa	Harrison	£235.67

Table 7

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
1	1	Simpson	Sofa	Harison	£ 235.67
2	1	Johnson	Chair	Harrison	£ 453.78
4	2	Jones	Suite	Harisonn	£3421.00
6	1	Simpson	Sofa	Harrison	£ 235.67
7	1	Jones	Bed	Ford	£ 453.00

Table 8

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
213	3	Williams	Suite	Harisonn	£3421.00
216	2	McGreggor	Bed	Ford	£ 453.00
217	1	Williams	Sofa	Harrison	£ 235.67
218	3	Aitken	Sofa	Harison	£ 235.67
225	2	Aitken	Chair	Harrison	£ 453.78

Table 9

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
3	2	Smith	Stool	Ford	£ 82.78
5	3	Smith	Sofa	Harrison	£235.67

Table 10

SaleNo	EmployeeNo	Customer	Item	Supplier	Amount
3	2	Smith	Stool	Ford	£ 82.78
5	3	Smith	Sofa	Harrison	£235.67

Restrict (aka Select)

Restrict simply extracts records from a table. Thus, if we perform a restriction on the table SALES where (Customer) = "Simpson", the result would be as shown in *Table 4*.

Union

Union creates a new table by adding the records of one table to those of another. Clearly, for this to work well it is essential that the tables have the same structure. The union of tables *Sales* and *Employees* is unimaginable because the two tables are clearly very different in structure. In order for tables to be "Union Compatible", they must have the same number of fields, and the fields must be of the same data type and size. The tables (*Sales* and *Sales2*) are "union compatible" and the result would be as shown in *Table 5*.

Note that the two records shown in *Table 6* were shared by the two tables, but have appeared only once each in the ANSWER (*Table 5)*. The order in which records appear in the result of a union is unimportant, but duplicate records are eliminated.

Difference

The difference of two tables is a third table which contains the records which appear in the first but *not* in the second. The tables concerned must be union compatible. Thus the difference of SALES (*Table 2*) and SALES2 (*Table3*) is shown in *Table 7*.

Note that, unlike Union, the order of the tables is vital. Thus the difference of SALES2 and SALES is not the same *(Table 8*). However, the records that are "missing" from the two ANSWER tables are the same *(Table 9*). That is, in both cases it is the records that are common to both of the tables involved in the difference operation.

Intersection

The intersection of two tables is a third table which contains the records common to both of them. Thus, the intersection of SALES and SALES2 is shown in *Table 10*.

Unlike the difference operation, the order of the tables is unimportant, but, of course, the two tables must be union compatible. ▣

PCW Contacts

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column. He's available on **m.whitehorn@dundee.ac.uk**