Ringing the changes

Yet another phone number change rankles with Mark Whitehorn until he gets code-crunching.

They are at it again. I can't believe it. They started in 1990, had another try in 1995 and now we are about to endure Round Three. We are talking telephone numbers here; changing the dialling codes in the UK is clearly a habit-forming occupation for Oftel.

The good news is that we are moving towards a standard dialling code pattern of three + eight; three numbers for the area code and eight for the local. The bad is that this move is being performed as a series of changes, so the current round won't be the last. Oftel claims to have consulted the 'business world' and it says that 'we' asked the changes to be phased in this way. I can only assume that the consultation process didn't include the DBAs (DataBase Administrators) who are required to actually implement the changes in corporate databases...

On the bright side, the changes which will have to be made are excellent examples of a more general requirement: 'How do you perform "batch changes" to the data in a database?' So, rather than just show you how to change phone numbers, we'll look at several strategies for making changes and then illustrate the solution to this particular problem using two of them.

1 Search and replace

Most PC-based databases, such as Access, have a Search and Replace facility. These can be really useful, but are frowned upon by serious DBAs, partly because said DBAs are data snobs, but mainly because they're obsessed with keeping the data in a database secure. If an S&R operation fails halfway through, the database could be left in an inconsistent state. In addition, if you are using a multi-table database, you risk doing serious damage to the between-table integrity of the data. So S&R is a great tool for simple jobs on single table databases – but it is to be used with caution.

In the case of the phone code changes it is an inappropriate tool because the changes are more complex than simply 'Look for 01247 2 and replace it with (028) 912'. To do the job properly we need to alter the spacing of the numbers as well.

The downside is that we need 143 hand-crafted update statements

2 Update queries

The update query is the preferred tool for performing multiple changes to records in a database. You create a query (using either SQL or a query builder) which essentially says: 'Find all of the records (entries in the database table) that match these criteria and then change the value of this field to that value.'

'Ah!' you're thinking, 'just like S&R'. True, except that the matching criteria for an update query can be more complex and it can implement more complicated changes to the record. In addition, an update query has the advantage of being a so-called 'set' operation, meaning that it is automatically applied to all of the records in the table. Set operations are generally preferred as a method of altering data because they are less likely to cause data integrity problems.

So, we can implement the phone code changes using simple update queries. The downside is that there are 143 changes to be made, so we need 143 hand-crafted update statements.

BlockOfNumbers : Table

OldNumber	NewCode	NewNumber
01203 XXXXXX	(024) 76XX XXXX	Z
01222 XXXXXX	(029) 20XX XXXX	Z
01232 XXXXXX	(028) 90XX XXXX	Z
01238 XXXXXX	(028) 97XX XXXX	Z
01247 2XXXXX	(028) 912X XXXX	Z
01247 4XXXXX	(028) 914X XXXX	Z
01247 5XXXXX	(028) 915X XXXX	Z
01247 8XXXXX	(028) 918X XXXX	Z
012477 XXXXX	(028) 427X XXXX	Z
01265 2XXXXX	(028) 702X XXXX	Z
01265 3XXXXX	(028) 703X XXXX	Z
01265 4XXXXX	(028) 704X XXXX	Z
01265 5XXXXX	(028) 705X XXXX	Z
01265 8XXXXX	(028) 708X XXXX	Z
01265 XXXXX	(028) 703X XXXX	Z
012656 XXXXX	(028) 276X XXXX	Z
012657 XXXXX	(028) 207X XXXX	Z
01266 3XXXXX	(028) 253X XXXX	Z
01266 4XXXXX	(028) 254X XXXX	Z
01266 6XXXXX	(028) 256X XXXX	Z
01266 8XXXXX	(028) 258X XXXX	Z
01266 XXXXX	(028) 256X XXXX	Z
012665 XXXXX	(028) 295X XXXX	Z
012667 XXXXX	(028) 217X XXXX	Z
01365 3XXXXX	(028) 663X XXXX	Z
01365 4XXXXX	(028) 664X XXXX	Z
013655 XXXXX	(028) 895X XXXX	Z
013656 XXXXX	(028) 686X XXXX	Z
013657 XXXXX	(028) 677X XXXX	Z
01396 5XXXXX	(028) 445X XXXX	Z
01396 6XXXXX	(028) 446X XXXX	Z
01396 8XXXXX	(028) 448X XXXX	Z
013967 XXXXX	(028) 437X XXXX	Z

Record: 1 of 143

◀FIG 1 THE SAMPLE DATABASE – THE Z CHARACTERS FLAG ANY UNALTERED NUMBERS

3 Create a lookup table and use a single update query to make the changes

The lookup table contains the old and new codes. The update query joins this table to the table of phone numbers, matches old to new, and makes the changes. The beauty of this method is twofold.

- It is highly adaptable for any further uses – you just need to change the values in the lookup table.
- You need, in theory at least, just one update query, not 143.

In practise, it isn't quite this simple because the changes to the phone numbers we need to make fall into 12 groups. Each group needs to be handled differently and you need one lookup table and update query for each group.

4 Use a programming language

Most database engines, as well as allowing SQL set operations to be performed, also have a programming language that can perform an update on the first record in the table. Once that record has been changed, it turns its attention to the next and so on down the table. This method of altering data is less secure than using set operations.

Choices, choices

So, which do we choose to demonstrate for your edification? Options three and

four are the most attractive. However, if we go for four we know that purists will complain that we should be using three. On the other hand, option three is more complex and probably overly so for many databases. So, we have solved the problem both ways and you can choose the one you prefer.

Making the changes

Our sample database [Fig 1] has a table called BLOCKOFNUMBERS. OldNumber contains an example of an old-style number with X replacing the non-code part of the number. NewCode contains the replacement code complete with Xs in the correct place. NewNumber currently contains a Z character. The Z flags the fact that the number has not yet been altered.

Option 3 – Using update queries

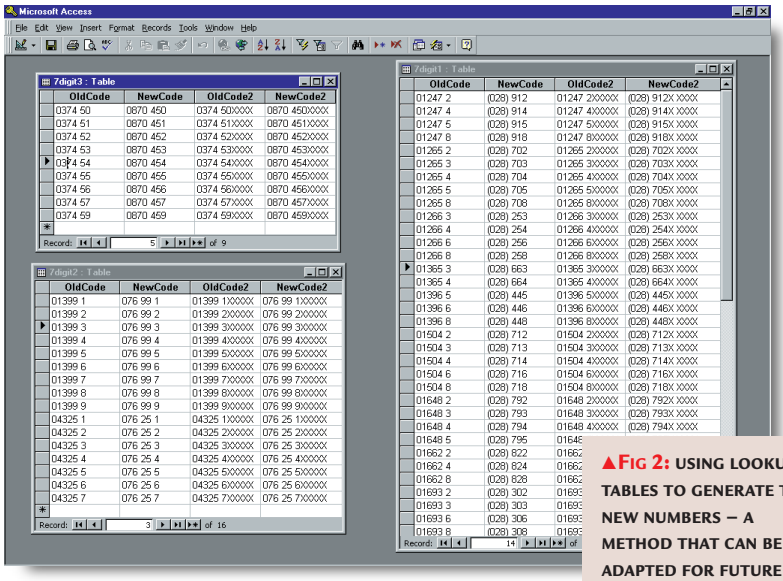
There are also 12 lookup tables with names such as 7digit3 [Fig 2]. Each lookup table contains a set of codes which need to be modified in the same logical way. The first numeral of the name (in this case 7) indicates the number of characters (digits and spaces) in the old code.

Each lookup table has a matching query, in this case Implement7digit3. When this query is run, it joins the lookup table to BLOCKOFNUMBERS and, where the old codes match, it generates a new number and places it in BLOCKOFNUMBERS. (In fact, this is a marginal lie. The query actually joins Implement7digit3 to a query called SevenDigit which is, in turn, based directly on BLOCKOFNUMBERS. However, all SevenDigit does is to make it easier to create the join to the first seven digits of the old phone number in BLOCKOFNUMBERS.)

This may sound complex but in practice it is easy to see what it does. Fig 3 shows the BLOCKOFNUMBERS table after the query Implement7digit3 has been run. Note that only some of the seven-digit codes have been replaced. If you then run Implement7digit2 and Implement7digit1, all of the seven-digit codes will have been altered. If you then run the remaining nine queries that start with the word Implement, all of the new numbers will have been generated. You must run the queries in descending order – that is, starting with 7digit3 and ending with 4digit1.

This sounds odd, but consider the following: 01265 2XXXXX becomes (028) 702X XXXX whereas 01265 XXXXX becomes (028) 703X XXXX.

Since both of these changes start with the same five numbers, when we run a query that looks for the 01265 code (which happens to be the 5digit2 one) we are in danger of changing the 01265 2 one incorrectly. However, this number is also found by the 7digit1 query which will change it correctly. Once a number has been changed, the queries won't change it again (by using that Z flag). So, by running the queries in descending order, we manage to change the numbers in the correct order.



▲FIG 2: USING LOOKUP TABLES TO GENERATE THE NEW NUMBERS – A METHOD THAT CAN BE ADAPTED FOR FUTURE USE

Option 4 – Using the code

Unfortunately, writing this code is definitely the more time-consuming option, but then it is very easy to run! You simply open the form called Generate-NewNumbersInBLOCKOFNUMBERS and press the button. This fires a big block of code which marches along, record by record, making the changes.

The button fires a big block of code which marches along making changes

Will this work for your database?

You won't be able to take the code presented on our cover CD and simply open the file and use it. For a start, there are many ways in which you could be storing telephone numbers currently – perhaps you store the area codes in a separate field.

In addition, both of the methods illustrated here will only work if the format of your current numbers exactly matches the format shown in the worksheet provided by The National Code & Number Change Programme. For example, they show the format for Coventry numbers as 01203 XXXXXX. However, if yours are stored as 01203 XXX XXX then our code won't convert the

BlockOfNumbers : Table

OldNumber	NewCode	NewNumber
01868 XXXXXX	(028) 87XX XXXX	Z
01893 XXXXXX	076 93 XXXXXX	Z
01960 XXXXXX	(028) 93XX XXXX	Z
0321 XXXXXX	0808 0XXXXXX	Z
0345 XXXXXX	0845 7XXXXXX	Z
0370 XXXXXX	077 70 XXXXXX	Z
0374 50XXXX	0870 450XXXX	0870 450XXXX
0374 51XXXX	0870 451XXXX	0870 451XXXX
0374 52XXXX	0870 452XXXX	0870 452XXXX
0374 53XXXX	0870 453XXXX	0870 453XXXX
0374 54XXXX	0870 454XXXX	0870 454XXXX
0374 55XXXX	0870 455XXXX	0870 455XXXX
0374 56XXXX	0870 456XXXX	0870 456XXXX
0374 57XXXX	0870 457XXXX	0870 457XXXX
0374 59XXXX	0870 459XXXX	0870 459XXXX
0374 XXXXXX	077 74 XXXXXX	Z
0378 XXXXXX	077 78 XXXXXX	Z
0385 XXXXXX	077 85 XXXXXX	Z
0401 XXXXXX	077 01 XXXXXX	Z
0402 XXXXXX	077 02 XXXXXX	Z
0403 XXXXXX	077 03 XXXXXX	Z
0410 XXXXXX	077 10 XXXXXX	Z
0411 XXXXXX	077 11 XXXXXX	Z
0421 XXXXXX	077 21 XXX	
04325 1XXXXX	076 25 1XX	

▲FIG 3: THE QUERY IMPLEMENT7DIGIT3 ONLY DEALS WITH PART OF THE PROBLEM, HENCE SOME CODES ARE LEFT UNCHANGED

ALL IN THE TIMING

The implementation of this round of changes is far from simple. Oftel is providing for parallel running (the old codes and new will both work for a time) but, horribly, the parallel running 'window' varies. Thus, for example, the old and new Coventry numbers will both work from 1 June 1999 until 19 August 2000, whereas the 01893 'find-me-anywhere' code runs in parallel with its new version (076 93) from 30 September 1999 until 28 April 2001.

As if this wasn't bad enough, these parallel-running window rules apply only to long-distance numbers; for local numbers the story is different. I quote: 'As part of the above changes, there will a "flash change" early on the morning of Saturday, 22 April 2000. Up to that time, the old local number will be used for local dialling, after that time only the new eight-digit local number must be used.'

So, how do you decide when to make the changes to your database?

A really important point is that there are no changes that start after 30 September 1999 and none that finish before 5 August 2000. This is significant because it means that there is a 'meta' window of opportunity for change between these two dates. If you implement the changes to your database between these dates your users should be able to use all the old numbers until you make the change, and all of the new numbers after you implement the changes. Even better, if you make the changes 'early on the morning of Saturday, 22 April 2000' then the local vs long-distance number difficulty shouldn't be a problem either. So this is the best date to implement the changes. However, you will, of course, need to do extensive testing (on copies of the data!) before this date.

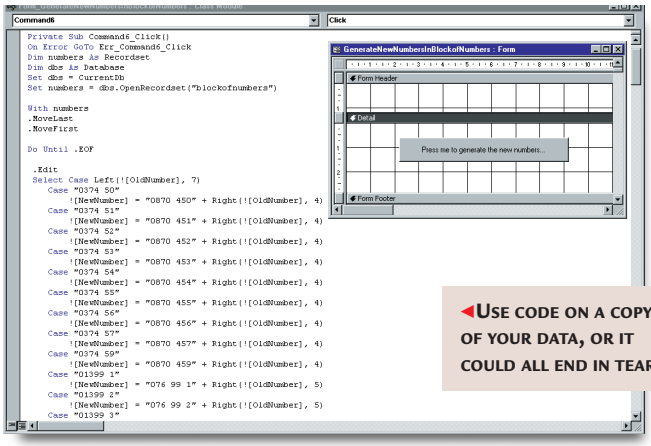
The documentation provided by Oftel and The National Code & Number Change Programme about the changes is available on www.numberchange.org.

The information at numberchange.org carries health warnings such as: 'Disclaimer: While every effort has been made to ensure that the information supplied above is correct at the time of writing, readers must be aware that there may be errors and omissions. The National Code & Number Change Programme reserves the right to revise this document without notice.'

In this article you are reading an interpretation made from a series of documents that is itself covered with health warnings. Clearly we cannot accept any responsibility for any errors or omissions made by that site or by us. This set of changes is relatively complex; don't try to make changes to a serious database without reading for yourself the documentation provided at the website above.

These riders aside, we have tried to make sure that the advice that we give is as accurate and as helpful as possible.

Can it all be automated? According to the Excel worksheet made available on the www.numberchange.org website, there are 188 changes. However, 45 of these are either brand new (hence no change is required) or are (090) numbers. According to The National Code & Number Change Programme: 'There is no fixed migration mapping from old to new numbers in the Premium Rate (090) range. Details on PRS migration arrangements will be made where required.' That leaves us with 143 code changes that we can automate, all of which are covered by this article.



USE CODE ON A COPY OF YOUR DATA, OR IT COULD ALL END IN TEARS

numbers properly.

You have two choices. You can modify your numbers to suit the 'approved' format before you start, or modify the code we provide to match your numbers. The latter is likely to be much easier, and is why you will probably want to alter our code before using it.

Another point to bear in mind is that the validation rules you may have set up in your existing database (perhaps it rejects numbers that don't start with 01) may have to be changed.

Finally, please experiment on a copy of your data until you are absolutely sure you know what you are doing!

PCW CONTACTS

Mark Whitehorn welcomes your feedback. Contact him via the PCW editorial office or email database@pcw.co.uk