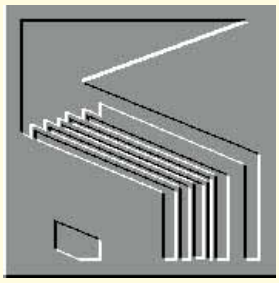

This **Codd** be the last time

Mark Whitehorn **wraps up the last of the list of suggested rules for modern, PC-based RDBMSs. There's Postcode, too, in shareware and full versions to help you get addressed correctly.**

Rule 10. The RDBMS must have a comprehensive control language (for example, SQL).

In order to act as a front-end to a database server, PC-based RDBMSs must have a control language and SQL is the obvious choice. Yes, I know SQL is flawed and that it is incomplete; I know it has many variations; but at least it is well established.

In addition, an RDBMS which has any pretensions towards use for serious work needs its own internal control language. While SQL can be squirted to remote database servers, the internal language is used to drive the interface, link forms and performing data validation. Whether this language is based on Basic, Pascal, C or any other reasonably common language doesn't matter too much as long as it is reasonably comprehensive. Support for macros is not, in my opinion, an acceptable substitute unless the DBMS is going to be used only for very simple databases.

Simple tasks

11. In addition, it must have a GUI interface which allows end-users to perform simple tasks like querying, reporting, etc.

The underlying language is essential for serious work, but who wants to code the bread and butter work (forms, queries and so on) by hand? GUIs have proved themselves remarkably well-suited to the task. I must admit to a personal liking for systems (like dBase for Windows) which allow you to use a GUI to build a form, say, but can then use your work to generate a code description of the form which you can then hand-tweak.

Results

12. The results of queries (answer tables) should, whenever possible, be editable.

This is, essentially, as Codd's Rule 6: "All views that are theoretically updatable are also updatable by the system". A "view" is essentially the same as an answer table.

As discussed in an earlier issue, actually determining whether each and every view is updatable has been shown to be impossible. However, it is easy to decide about most views and the RDBMS should allow us to update all of those views where it is clearly safe to do this. Of the current crop of RDBMSs, only Access and Paradox provide editable answer tables and of these Access has the more extensive implementation.

Multiple records

13. It must be possible to alter multiple records with a single command.

And this is essentially Codd's Rule 7: the "High-level insert, update and delete" rule. Most of today's PC-based RDBMSs provide this facility though some of the more simple DBMSs don't.

Cascade

14. The RDBMS must support referential integrity, with cascade update, cascade delete, etc.

Primary keys help to ensure that the data within tables is internally consistent. By something of the same token, Referential Integrity is a way of ensuring that the logical relationships between tables are maintained.

Given the two tables in *Fig 1*, it is clear that the numbers in PATIENTS.[Gp No]

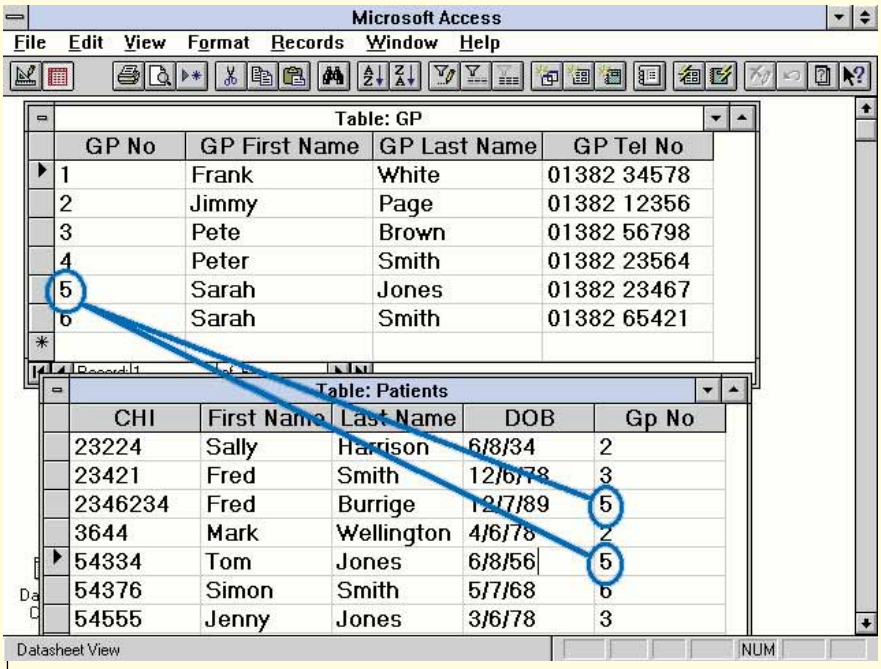

Fig 1 *Showing the association between GP.[GP No] and Patients.[Gp No]*

refer to those in GP.[GP No]. So both Fred Burrige and Tom Jones share Sarah Jones (no relation, at least in the kinship sense) as their GP. Considering that the GP table only contains six GPs, numbered one to six, it would be meaningless to place the number seven in the PATIENTS.[Gp No] field; unless of course we added another GP with that number to the GP table. Referential Integrity provides this check, and refuses to allow an entry in PATIENTS.[Gp No] unless there is a corresponding value in GP.[GP No]. Referential Integrity will also forbid the deletion of a GP record if it refers to one or more PATIENT records which exist. To allow this deletion would leave "orphaned" records pointing to a non-existent GP.

• *Cascade Delete*

So, what of cascade delete? This is an option for referential integrity, an add-on if you like, which says that if you do delete an existing GP record, referential integrity is maintained by deleting all of the PATIENT records which refer to that GP record. Typically, if cascade delete had been set for this join, when you tried to delete the GP record the system would warn you that Patient records would be deleted as well, and offer you the choice to proceed or abort the process.

You, as the database designer, may decide that cascade delete is inappropriate in this case (which it clearly is) but it can be really useful in other circumstances: in order/sub-order systems, for instance. RDBMSs should offer it to us as an option so that we can decide if and when to apply it.

• *Cascade Update*

"Standard" Referential Integrity will also not allow you to delete or change the value in GP.[GP No] when that value is referenced by records in PATIENTS. Cascade Update, like Cascade Delete, is an optional extra which can be added to Referential Integrity. It will allow you to change the value in GP.[GP No] and it will then seek out and update all the values in PATIENTS.[Gp No]. This will be entirely inappropriate for some applications while for others it will be an essential requirement. On those occasions, you will be pleased that you chose an RDBMS which supports it, and so will your employers when you casually let them know all about the excellent choice you made.

Support and sort

15. The RDBMS must support the maintenance of indices as well as sorting.

Human beings like their data to be sorted

Postcode — an interesting new toy

Postcode, from AFD Software, is a system which looks up addresses from postcodes. Suppose that you are developing an application in, say, Delphi. Your client wants to be able to type in a postcode and have the address magically appear on screen (great for telesales operations). As the programmer, you could, over the next three years or so, dutifully type in all of the known addresses and postcodes… or you can simply plug the Delphi version of Postcode into your application. Sample code is already provided for Visual Basic (for DOS or Windows), MS Access, Delphi, Paradox and others. If you want Postcode for other, less well known systems, AFD says it will write sample code for you. The datafile is about 18Mb, which includes the indices which reduce typical search times to less than one second.

Nildram Software has been appointed as distributor for the shareware release of Postcode. It ships with a DOS TSR version, a Windows version supporting DDE and direct pasting into applications, plus programming interfaces for the languages described above. Due to licensing restrictions the shareware version of Postcode doesn't supply addresses down to street-level detail — just towns and counties. Happily, for those intending to download it via a modem, this also reduces the size of the data file to about 600Kb. This version can be registered for just £42.50 (plus VAT) with no other licensing costs. If you want the full version which works down to the street level, it is still only a modest £99 (plus VAT) for the software and an annual licence fee of £55.

A Unix version of Postcode is currently under joint development between Nildram Software and AFD, and is expected to ship in the early Autumn. Pricing has yet to be announced but should be as competitive as the DOS and Windows version.

You can try out Postcode on-line at http://www.nildram.co.uk/nildram/postcode.html. On this Web page you can test out the Unix version of the software while on-line, download a copy of the shareware version, and even register it there and then. For users who only have FTP access, a shareware version of Postcode can be obtained from ftp://ftp.nildram.co.uk/pub/nildram/afdpost.zip. Nildram also runs a BBS where users can download the software for free, on 01442 891109 with your modem for access.

either numerically or alphabetically. But records in a table are rarely entered in alphabetical order so if we look at the table "in the raw" the records are usually not in the order we want. The answer is to get the RDBMS to order them for us.

One way in which the RDBMS can do this is to sort the records by physically moving them around in the table — that is, by changing their position within the file on disk. Given a file of any size this process is horribly slow because of the disk I/O involved. In addition, as soon as more records are added, the entire process has to be redone. Worse, what happens if you want to see the data sorted in different ways? You might sometimes, for instance, want the data sorted alphabetically by name, yet at other times by telephone number. If this is done by physical sorting on the disk, the RDBMS will have to maintain two tables, each sorted by a different field. Clearly this is wildly inefficient.

In the light of this problem, indexing was born. An index is essentially a list of numerical values which gives the order of the records when they are sorted on a particular field. For example, in *Fig 2* you can see the GP table in its original state: the records are in "entry" order. If we generated an index on the [GP Last Name] field, it might look like the table on the right called Last Name Index. This shows that given the required sort, the last record would be [GP No] = 1, Frank White. To save you the trouble of verifying this, the table shown lower left contains the same data actually sorted by last name. Another "index" is also shown (bottom right) for Tel. No — if you are feeling dedicated you can check to see whether it is correct.

Fig 2 *On the right are two tables mimicking the way in which indices work. The upper one shows how an index based on how [GP Last Name] would look; the lower one represents an index based on [GP Tel No]*

Compared to physically moving records around on disk, indices are much easier and faster to generate and maintain. In addition, you can have as many indices as there are fields in your table without wasting too much disk space.

It is worth stressing that indices are maintained internally by the RDBMS — they don't appear as small tables, as I have shown here. I constructed these small tables by hand simply to illustrate the principle — you won't find them appearing like this if you index a table.

Quick queries

Indices are really useful for sorting data for human consumption but they are also invaluable for speeding up processes like querying. A word of caution however: despite their efficiency, indices do take *some* processing power to maintain and shouldn't be used indiscriminately.

So how do you know which fields you need to index? The answer to this question comes under a more general heading, namely "How do I speed up my database?" Readers have asked me to discuss this more general topic, so I will return to the subject of indices in a couple of issues' time.

For the moment, the bottom line is that any RDBMS worth its salt must allow you to mark one or more fields as indexed. This is usually done during table design, and once you have done that, the RDBMS should construct and maintain the indices transparently.

Tips & Tricks

Tips and tricks are proving popular, so let's have some more — I'll supply the ones I find and if you have any good ones, for whatever RDBMS, please send them in.

Grid tip

From Shane Devenshire, Walnut Creek, California: *"Access uses a Snap to Grid feature to help align controls on forms and reports. Unfortunately, it may appear not to have any effect when you first begin to use Access because the default is 64 gridlines per inch. At this resolution Access chooses not to display the grid on screen (even if you have selected "Show Grid" to be on); and the impact of this very fine grid on the positioning controls is negligible.*

The spacing of the gridlines are Form Properties. You can set the spacing to less than 64 but you will find that you still can't see the grid until the Grid X and Grid Y properties are set to less than 18."

The screenshot alongside shows the property setting for a form and the relevant grid settings are visible. It also shows (on the right) the place where you can see and alter the "Show Grid" option. This is a property for the entire database so you set it by popping down the View menu, selecting Options and highlighting Form & Report Design. If you don't want the controls to line up with the grid, set the "Snap to Grid" option (found in the Format menu) to Off.

In my experience, Access 1.1 defaults to Grid X = 5 Grid Y = 5 for blank forms, and Grid X = 64 and Grid Y = 64 for wizard-generated forms. My copy of 2.0 defaults to Grid X = 10, Grid Y = 12 for both, but I haven't examined this extensively, so "your mileage may vary".

This take-home tip is excellent. Whichever version of Access you are using, be aware that you can alter the grid size, which in turn affects how controls line up on the form.

Broad search

In the very simple DBMS called Q&A you can use the search facility to find data in fields, but the search, by default, is exact. To find, say, all data containing the word "Penguin" use:

```
..Penguin..
```

which should find entries such as: "The Adventures of Penguin Penguinsson".

Setting the Form Properties in Access so that objects snap to grid

PCW Contacts

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

Nildram Software 01442 891331.
Fax 01442 890303.
Email sales@nildram.co.uk