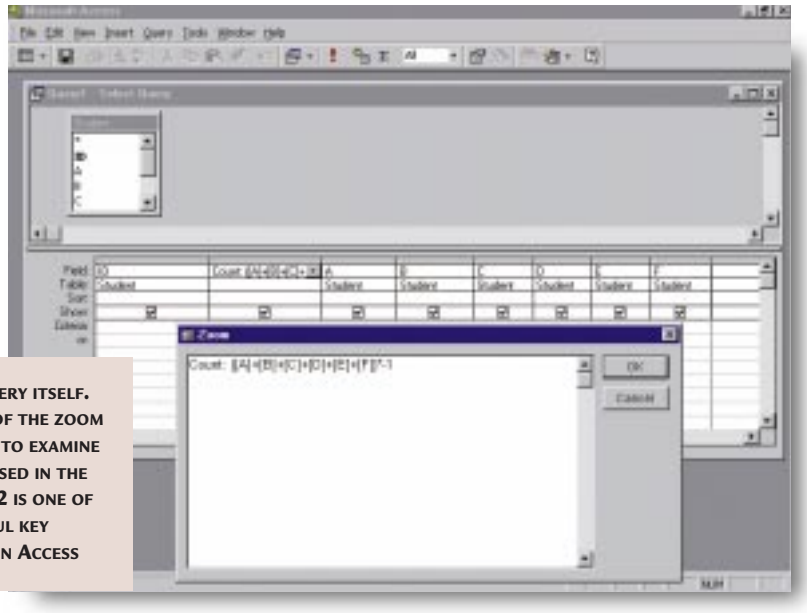
Off the shelf

Even small firms benefit from data warehousing. Mark Whitehorn explains.

Apologies — in the February column I wrote about data warehouses and promised more to follow. However, other database issues have distracted me in the interim. But no matter, here now is more on the subject of data warehouses. But first, I will quickly re-cap.

A data warehouse is an online, non-operational, de-normalised database (*That's a really intelligible definition, Whitehorn — Ed*). OK, these definitions may help:

- **Online** — users have immediate access to the database, as is normal for most databases.
- **Non-operational** — no-one is adding to, or altering the data in this database. Note the crucial difference in meaning between 'online' and 'operational'. People sometimes mix these two up and take 'online' to mean that the data can be altered by users.
- **De-normalised** — we typically normalise data in an operational database to ensure data integrity as users add and alter data. Since the data warehouse is non-operational you can optimise the structure for querying rather than update — but read on...



▶ FIG 1 THE QUERY ITSELF. NOTE THE USE OF THE ZOOM BOX (SHIFT F2) TO EXAMINE THE FORMULA USED IN THE QUERY. SHIFT F2 IS ONE OF THE MOST USEFUL KEY COMBINATIONS IN ACCESS

Users do not alter the data in a data warehouse. Instead, the data exists solely so that people can run queries against it. These queries are typically not seeking specific records within the data but are broad questions which are attempting to identify trends within it.

A data warehouse does not have to be a complex, expensive project

The data warehouse is a copy of the data in an operational database. Typically the information in a data warehouse is refreshed from the operational database at regular intervals of, say, once a week.

A data warehouse used to be the preserve of big companies, and data warehouse projects have become inextricably linked with complexity, expense and failure — there is an oft-quoted factoid that 70 percent of data warehouse projects fail. This may or may not be true, but the following scenario seems common.

Suppose a large company decides it needs a data warehouse. It has a huge volume of data, so the project is complex from the start. It requires big, expensive bits of hardware and software. The project is clearly going to be pricey so committees are put in place to set targets, monitor progress and ensure the success of the project. Sadly, this whole bureaucratic mechanism, designed to ensure success, can gum up the whole development in a morass of reports and meetings, stifling both innovation and progress. Eventually the project loses momentum and ends up dead in the water and the committees become nothing more than devices to reduce individual accountability.

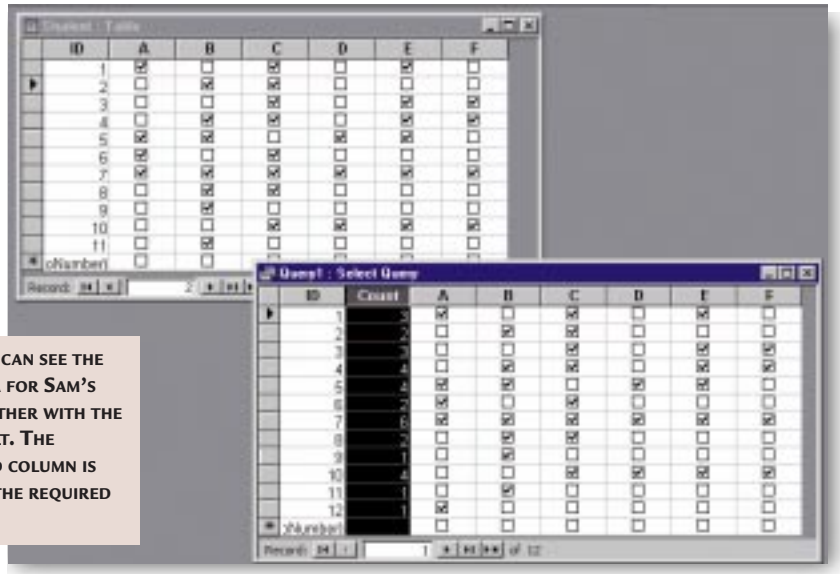
What has all this got to do with data warehouses and you? Well, hardware and software costs have plummeted. Small- to medium-sized companies have less data but in this competitive world no less a need to analyse trends. Creating an online, non-operational copy of your data on a separate machine is no longer expensive. If you simply do this and leave the structure exactly as it was in the operational database (i.e. normalised) you can, crucially, still separate the functions of updating the information from that of running large, complex queries against it. In other words, a data warehouse doesn't have to start out as a complex, expensive project. You can start simply. If it turns out to be useful and people begin to demand more functionality you can then look at the cost advantages of making the data warehouse more complex, which usually involves restructuring the data.

Even then, this restructuring doesn't have to be complex at first. For example, you have a database that stores details of all the orders that have been placed with your company over the past five years. The sort of queries that run against the data warehouse are rarely specific. For instance, 'Show me exactly what Fred Smith ordered on the 23/11/1995'. Instead, they often deal with summarised information such as 'How has the value of the average order varied with time?'

Suppose that your data warehouse is refreshed each weekend from the operational data. Once the copy has been made, you could run a series of

queries in the data warehouse which generate summaries of the information therein, and write the results to new tables within the data warehouse. Your users can then run queries against these summary tables, getting their answers more rapidly. Of course, deciding exactly what summary tables will be the most useful requires both skill and consultation with your users. However, this can be an iterative process while the data warehouse is already proving useful.

Clearly there is more to data warehousing than this and whole books have been written on the subject but this brings us very neatly to...



▶ FIG 2 YOU CAN SEE THE SAMPLE DATA FOR SAM'S TABLE, TOGETHER WITH THE QUERY RESULT. THE HIGHLIGHTED COLUMN IS DISPLAYING THE REQUIRED TOTAL

■ ...Counting the 'yes' men

Reader Sam Browne wants to know how to count the number of 'Yes' responses given by students.

Each response from the student is stored as one row in a table. Sam goes on to point out: 'There are six yes/no fields in the main student table, which represent six different selection criteria — let's call them A, B, C, D, E, F — and each record can have any combination. I want to include a simple calculation in a query which totals the number of criteria for each record.'

The following lines of SQL should do the trick:

```
SELECT Student.ID, ([A]+[B]↙
+[C]+[D]+[E]+[F])*-1 AS↙
 Count, Student.A, Student.↙
B, Student.C, Student.D,↙
 Student.E, Student.F
FROM Student;
```

(↙ Code string continues)

This essentially gives the instruction to total the values in A, B, C, D, E and F, then multiply the result by minus one and put the answer into a column called Count. Then show me the original columns, as well. See Figs 1&2.

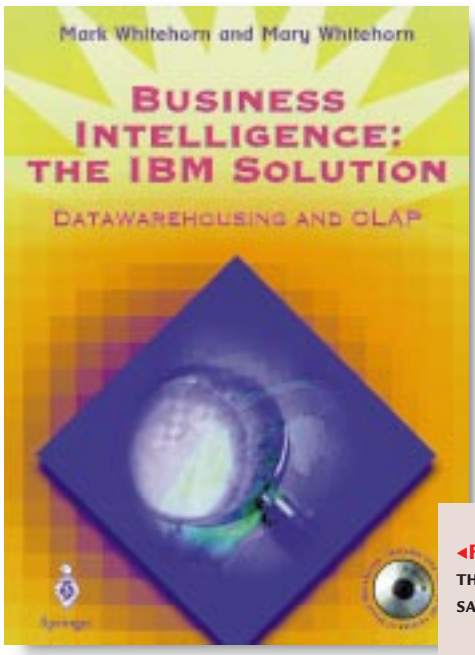
It relies on the fact that Access stores a 'Yes' as the value minus one, and a 'No' as a zero. Thus the query simply adds up the numbers and finally multiplies by minus one to make the count positive.

• *The sample database is on this month's cover-mounted CD-ROM.*

■ Books

For the past year, *Personal Computer World* has been kind enough to offer a special deal on the book which Bill Marklyn — the original development manager of Access — and I wrote about the relational model. This book is published by Springer and we were delighted to discover that it had become their best selling book in the UK during 1998.

Bill and I are currently working on two other books: one about using Access and the other about those areas of databases which fall outside the relational model, yet still need to be explained. Essentially this is a sequel (or, indeed, a SQL?) to the previous book *Inside Relational Databases* and will probably be called *Outside Relational Databases*. A tentative selection of topics includes data warehousing, OLAP, OLTP, client server, three-tier databases, row-level locking versus page locking, transaction control... and so the list goes on.



◀ FIG 3 MORE ON THIS SUBJECT BY THE SAME AUTHOR

And that is where you can help. We would be grateful for suggestions about topic areas which the readers of *PCW* would like to see covered. If you have any ideas, please email them to the usual address with the subject heading 'Book Suggestions'. Springer has offered five copies of the relational database book which will go to the five most useful/exhaustive/sensible suggestions — and the judges' decision is final!

In my spare time I am working on two more books with the other M.Whitehorn (my wife, Mary). One book is on upsizing from Access to SQL Server and driving SQL Server generally, while the other is about SQL Server for OLAP. In case this makes us sound Microsoft biased, we have also previously published two books on IBM's database and data warehousing products, one of which is shown here [Fig 3].

PCW CONTACTS

Mark Whitehorn can be contacted via the PCW editorial office (address, p14) or email database@pcw.co.uk