Patents impending

Two IBM patents, for data storage and classification, highlight the company's commitment to research. When the theory needs to become practice, IBM will be there, says Mark Whitehorn.

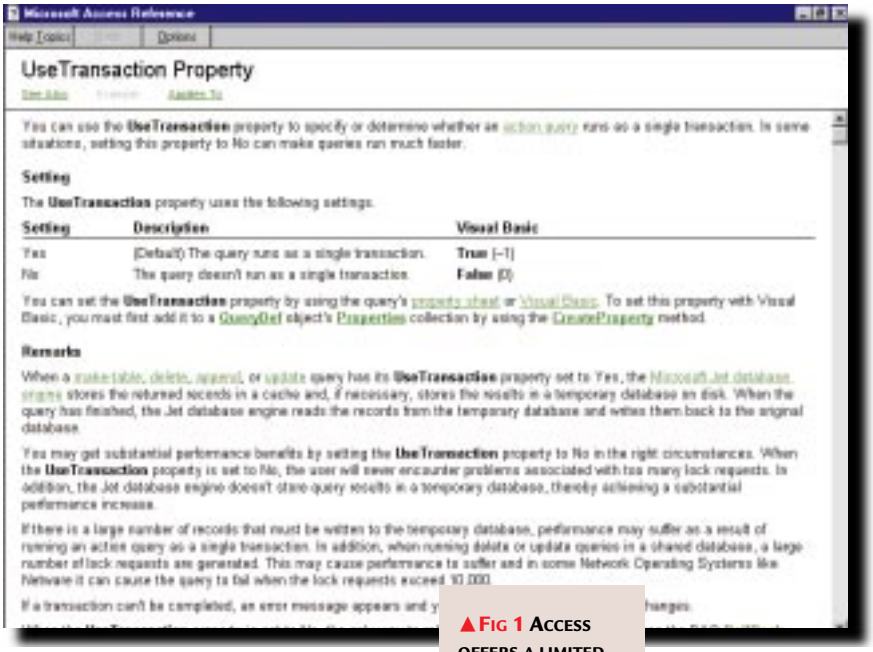
I was fortunate enough to spend some time at the IBM research lab in Santa Teresa recently. In an oblique way, the visit brought home to me how much Big Blue is prepared to invest in pure research.

I was in Ron Bingham's office talking about IBM's business intelligence tools when I noticed two patents hanging on his wall and idly enquired as to what they covered. The first turned out to be for a theoretical system that enables any and all data to be classified and stored.

"Surely you can't mean *all* data," I said. "How many objects can it classify?"

"Well," replied Ron (an IBM researcher), "more than there are fundamental particles in the observable universe."

The second patent described something which would allow such a storage/classification system to be implemented. It involved a hierarchy of servers that communicate and issue classifications. It also seems to allow for compression so that objects can be stored more efficiently. I say "seems to", because the patents were fairly mathematical and my brain couldn't cope with the details. In fact, I apologise to Ron if I have misrepresented them. The point is that IBM gave him time to work on a topic that is currently totally theoretical. One day, someone will need this kind of ability in their database and IBM will be there, patent at the ready.



▲ FIG 1 ACCESS OFFERS A LIMITED FORM OF TRANSACTION CONTROL. I KNOW OF A SYSTEM WHERE TRANSACTION LOGGING HAS BEEN IMPLEMENTED, BUT IT'S NOT A STANDARD FEATURE OF THE JET ENGINE

On the trail of transactions

And with that, we move elegantly from a highly theoretical subject with no practical application, to transactions, a theoretical subject with very immediate applications. So first, the theory, and then, why its application is so important.

Certain operations in more complex database applications are intimately tied together, at least in the logical sense. For example, Andy buys Sophie's car. The price of the car is removed from Andy's bank account. Then it is placed in Sophie's account.

If the system performing these actions should crash and only part of the process were completed, then Sophie is going to be very upset (and neither will Andy be best pleased). So, what we do is "tell" the DBMS that these two operations form an entity called a "transaction". The DBMS is told to treat this transaction as one object. Either the entire transaction must succeed, or it must all fail; there must be no half measures, even if the database crashes halfway through the operation.

Roll call

In order to provide this facility the DBMS must, before it starts to carry out the operations in the transaction, write the details of what it is about to do to a file on disk. Then, if the database crashes, when it comes back up it can look at the file and "roll back" the unfinished transactions. (The same can happen if the workstation that initiates the transaction crashes.) The ability to manage transactions in this way can be called "transaction control".

The step from here to "transaction logging" is simple. Instead of discarding the information about each transaction once it is complete, the information is retained in a log file. Indeed, every

TO BE PRECISE

I recently looked at converting characters using the built-in upper-case function. Anthony Blyth <Anthony.R.Blyth@lloyds.com> sent a function (see code.txt on our cover CD-ROM) based on the ASCII codes allocated to letters. This method requires a more intimate understanding of ASCII codes but it enables precise manipulations and conversions. It's trickier, but gives greater control.

operation carried out against the data in the database is logged in this file.

Now, imagine a system that's backed up at midnight on Sunday and that work proceeds as normal until 3.27pm on Monday when a crash'n'trash event occurs. You restore the database to the state at the last backup, and then "roll forward" through all the completed operations in the transaction log, thus bringing the database back to the state it was in just before the crash. Incomplete transactions are, of course, not rolled forward. The restoration and rolling forward can be handled automatically by most client-server DBMSs.

Transaction control and transaction logging work to ensure the integrity of your data in an uncertain world.

The moral of the story? "Take nothing for granted" — except recently, I did.

The sting

While in the US I was asked to attend a presentation, given by the vendor of a financial application, to a prospective client. The package was a client-server system with clients running Windows 95 and the back-end running on Unix.

The motivation for purchasing the software came from the financial arm of the organisation, which was satisfied with the way it performed the required financial transactions. I was happy to accept their opinion, accounting procedures being outside my remit.

As for the underlying structure of the RDBMS, I initially took it as read that the proprietary software sat atop WUT (well understood technology) — something like Oracle, or DB2. It took a regrettably long time for me to twig that not only was the software proprietary, but also that the data was stored in a proprietary flat-file system and that a proprietary program was used on the Unix box to access and control that data.

Out of control

At this point, even I could no longer fail to hear the tolling of alarm bells. After all, although it's quite possible for a group of dedicated programmers working for a small company to create an entire DBMS, it is unlikely that they will do better than the entire might of Oracle, or IBM, or Microsoft. So I had to backtrack on the questions I was putting forward and start asking about fundamentals like transaction control. The answer was simple: there was none. If a workstation hung during order processing, the entire system had to be closed and utilities run to check the integrity of the data files.

There was no transaction log, so even if the system was backed up once a night, any orders created after the backup and prior to a system crash would have to be re-entered.

From bad to worse

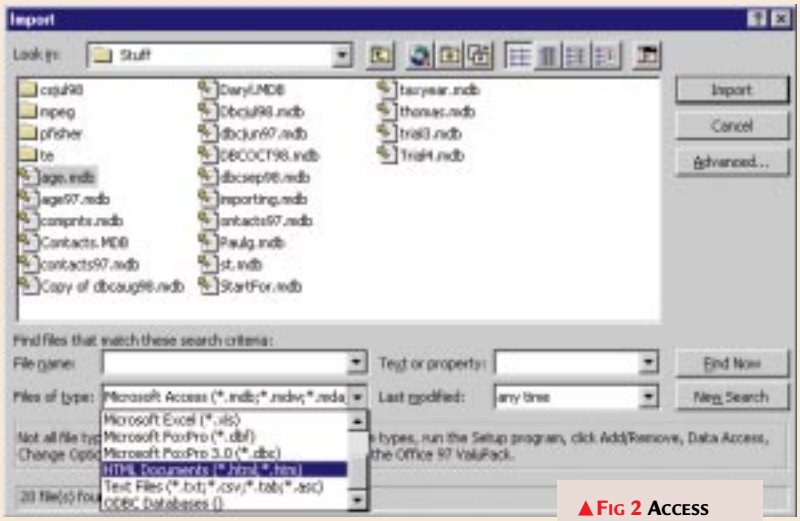
This is bad, and it happened to be worse in this particular case. Creating an order automatically printed a paper copy of that order, complete with an order number that was issued sequentially to each as it was created. These paper copies were ready for immediate despatch to suppliers by snail-mail.

It follows from all of this that to re-create the order information which may already have been sent to suppliers, it is necessary to re-enter the data in exactly the same order as was used prior to the crash. If not, re-generated orders won't bear the same order number as the original. The bad news is that re-entering the data in exactly the same order is essentially impossible, given a multi-user system.

This is not a diatribe against small outfits producing proprietary software — far from it. The imagination and flair of such companies adds enormous value to the existing offerings of big corporations. Small companies can react quickly, filling niches and producing brilliant products. However, there is a strong argument for basing such products on existing DBMS technology and putting programmer effort into making it a superlatively interfaced and highly customised "accountant's friend".

Access has some degree of transaction control [Fig 1] but if you want proper transaction control as described here, you would be well advised to look to a client-server system.

DOS-TO-ACCESS TRANSFER



▲ **Fig 2 Access imports from sources including dBase, Excel, FoxPro, text (CSV and straight) and ODBC**

Thorkil Mailand <thorkil.mailand@has.dk> from Denmark has asked about transferring data from an older DOS program to Access. In general, the best approach is to export the data from that package in a format which Access can import [Fig 2].

PCW CONTACTS

Mark Whitehorn *welcomes readers' suggestions and feedback for the Databases column. He can be contacted via the PCW editorial office (address, p10) or email* **database@pcw.co.uk**