IBM'S DATABASE2 VERSUS CULLINET'S IDMS/R

By James Bradley

Two of the most important data base management systems on the market today are IBM's DATABASE2 and Cullinet Corporation's IDMS/R. A little history helps to put these two systems in perspective.

In the 1970s two types of data base system dominated commerce. These were **hierarchical** and **CODASYL network** systems. CODASYL systems were based on recommendations published by CODASYL, the voluntary body that originated COBOL.

By far the most common hierarchical system was IBM's IMS, and the most common CODASYL system was Cullinet's IDMS, which later evolved into IDMS/R. During that period it was common to refer to CODASYL systems as **network** systems, to distinguish them from hierarchical systems.

The term network is still applied to CODASYL systems to this day. Unfortunately, this can be very confusing, for it does not help distinguish CODASYL systems from relational systems, like DATABASE2, which began to become important in the 1980s. The problem is that relational systems are fundamentally also network systems.

So perhaps the best way to compare DATABASE2 and IDMS/R is to begin with what they have in common. As we have seen, both systems are fundamentally network systems. What exactly does this mean? A data base is fundamentally a collection of files that are related. With a relational data base the files, as seen by users,

are rather restricted in format - no variable length records and no duplicate records are allowed. In more technical terms, the files are of a restricted type known as **relations**. With a CODASYL system, the files are not restricted to relations, and variable length records are allowed, although they are not common; thus, in practice, most files in a CODASYL data base will often qualify as relations. Therefore, from a practical point of view, we can often ignore the fact that relational data bases are made up of a more restricted kind of file than CODASYL data bases.

We have said that the files of both types of data base form a network. We mean here a network as opposed to a hierarchy. If the files of a data base form a hierarchy, we have a pyramid structure, with one file at the top called the root file. Call this file A. At the next level down there could be files B and C. A will be the parent of B and also the parent of C. This means that for one A record there are many related B records, and also for one A record many related C records. B is the child of A and so is C. This means that for one B child record there is only one related (parent) A record, and for one C child record only one parent A record.

In the hierarchy, B will be the parent of some child files at the next level down, perhaps P, Q and R; similarly C may be the parent of child files W, and X, and so on.

In such a hierarchical structure, every file except the root file has a parent, or equivalently, every file has zero or one parent files. It was such structures that IBM's IMS was originally designed to manage.

Very simply, if a structure of related files does not form a hierarchy, it must form a network, for in a network, at least one

file will have more than one parent. Therefore any structure will form a network, if it does not already form a hierarchy. Both relational and CODASYL systems can handle just about any data base structure, and are therefore both network systems.

This last statement needs some qualifying, however. What connects the files in a network structure, and also in a hierarchical structure, is **relationships**. We saw that with the hierarchical structure, for one parent A record, there could be many child B records. The relationship between files A and B is **one-to-many**, or **parent-child**. One-to-many relationships are by far the most common in all data bases, and they commonly connect the files of both CODASYL and relational systems in a network.

But CODASYL systems are designed to handle only one-to-many relationships, and they do this by means of a construct called a **CODASYL set**, where each element of the set consists of a parent record together with its child records. Commonly, such sets are implemented by means of pointers embedded in the records of the files, as with IDMS/R. With relational systems the one-to-many relationships are handled by a variety of methods that may or may not involve pointers. DATABASE2 does permit the use of pointers, although these never have to be specified by the person defining the data base; in contrast, an IDMS/R CODASYL data base definition requires pointer specifications.

To summarize the discussion so far, what is common to relational systems like DATABASE2 and CODASYL systems like IDMS/R is (a) both system types can manage data bases that have a network structure, and (b) the files of the network are connected by one-

to-many relationships that are implemented differently in the two system types.

But there are other types of relationships besides the common one-to-many relationships. There are also **many-to-many** relationships. But both relational and CODASYL systems can handle these, since a many-to-many relationship always breaks down into a pair of one-to-many relationships.

However, a reasonably common but poorly understood type of relationship called a **co-relationship** can be handled easily by relational systems, but not at all by CODASYL systems. The problem with a co-relationship is that it has no one-to-many aspect that will allow the CODASYL set structure for one-to-many relationships to be used.

Relational systems handle relationships by equating field values in related files, so that essentially any kind of relationship can be handled, one way or another. Thus the network structure for a **relational** data base can have files connected by relationhips, such as co-relationships, that would not be permitted with the CODASYL approach. And incidently, although it is beyond the scope of this article, co-relationships are important, for recent research has shown convincingly that they are the source of the ubiquitous connection trap.

This brings us to the essence of the difference between CODASYL and relational data base systems. Even though both systems permit the management of network data bases, **the network data base in the relational case may be made of up files that are connected by a richer variety of relationships than in the CODASYL case.**

This richness, and the flexibility that results, has made

it possible to design non procedural languages of great power for manipulating relational data bases. The SQL language for DATABASE2, which has been widely implemented in other relational systems as well, is the best example. CODASYL systems have no language that can compare. However, to meet the competition, Cullinet has added a relational front end to its CODASYL IDMS system, calling the resulting hybrid system IDMS/R. However, IDMS/R does not permit the use of SQL, which is rapidly becoming the standard non procedural data base language.

[With a non procedural language, you specify the processing required, instead of constructing a routine to specify how the processing should be carried out. With relational systems the required processing routine is generated automatically from the specification, in SQL, of the processing required.]

If the relational front end of IDMS/R does not permit the use of SQL, what does it do? It permits the use of **views**, something that is easily possible with relational systems, but not CODASYL systems. Remember that systems like DATABASE2 permit use of the non procedural language SQL. With an SQL expression you can specify the construction and retrieval of what is essentially a new file formed from data in multiple files of the data base. If this new file is specified as a view (with SQL), it can then in turn be used for further manipulation by SQL. This facility can be very useful when a quite complex SQL expression is needed to construct the view, but only simple SQL expressions are needed afterwards to manipulate it. Without the view facility, complex SQL expressions would be needed with every manipulatiuon of the data involved.

IDMS/R provides two facilities for handling views. One is

the logical record facility (LRF) that permits a view to be formed from a CODASYL data base. The other is Automatic System Facility (ASF), which permits a data base to be defined with files that are relations, with no need for CODASYL set definitions for the one-to-many relationships. This type of ASF-relational data base is manipulated procedurally within a program, and can be manipulated non procedurally at a terminal, in a somewhat restricted fashion, by a language called OnLine Query.

In summary, IDMS/R is fundamentally a CODASYL system, with some facilities of a limited nature akin to those commonly found in relational systems. In contrast, DATABASE2 is close to being a true relational system, with all the flexibility that that entails. Nevertheless, both can handle a network structured data base whose files are connected by common one-to-many relationships.

It might look from this discussion that DATABASE2 is undisputedly the better system. But better is a subjective term, and we would be wise to ask better for what. There is no doubt that DATABASE2 rests on a superior foundation, whereas IDMS/R clearly rests on a more ad hoc foundation. But the superior foundation and powerful facilities in DATABASE2 take their toll when it comes to ordinary transaction processing. For processing transactions with a few data base files connected by the common one-to-many relationships, DATABASE2 is currently significantly slower than IDMS/R, mainly because of the sheer amount of DATABASE2 code that has to be executed per transaction. Thus for ordinary transaction processing with ordinary data bases, IDMS/R will do the job, and do it very well. The point about DATABASE2 is that it will do things that are either very difficult or even impossible with IDMS/R, such as non

procedural manipulation involving off-beat relationships. If that is the kind of thing you require then DATABASE2 is indeed the better system.

About the author

Dr. Bradley specializes in data base management at the University of Calgary. He is the author of **File & Data Base Techniques,** 1982, **Introduction to Data Base Management in Business**, 2nd Edition, 1987 (January), and **Case Studies in Business Data Bases** (fall 1987), all published by Holt, Rinehart & Winston, New York.