UNDERSTANDING RELATIONS

(Installment #4)

E. F. Codd

Q1 A data model in which functions play a central role appears to have the advantage of making algorithms and data more interchangeable. Why not replace the n-ary relations of the relational model by functions (some or all of which may be set-valued)?

A1 If you are trying to convert the relational model into a model for describing the various logical and physical data structures that are now in use, this might be an appropriate step to take. Let us assume, however, that your intent is to preserve the dual role of the relational model in data base systems architecture -- in the principal schema the role is to serve as a framework for data control (authorization control, access and update concurrency control, integrity, and recovery); in the user schema the role is to serve as a common, general purpose interface for users of many different types (not just programmers). In this case a strong argument against such a step is that either it destroys the symmetric exploitation property of the replaced relations or else it requires an absurdly redundant collection of functions.

To illustrate the problem consider the relation SUPPLY cited in Installment #3 (FDT, vol.6, no.2). A sample extension (or tabulation) appears below:

SUPPLY (SUPPLIER#	PART#	PROJECT#)
S1	P1	J1
S2	P1	J1
S1	P1	J2
S1	P2	J1
S2	P2	J2

One set-valued function that might replace this relation has SUPPLIER#s as domain together with a range whose elements are all possible subsets of pairs of PART#s and PROJECT#s. Its tabulation would appear as follows:

S-TO-PJ (S# P#J#)

S#	P#	J#
S1	P1	J1
	P1	J2
	P2	J1
S2	P1	J1
	P2	J2

This function is very pleasing if, in data base control and use, all you ever need are combinations of PART#s and PROJECT#s for a given SUPPLIER#. Unfortunately, this type of assumption is untenable in almost every data base, and is especially untenable in highly shared data bases. Each of the other five functions (P-TO-JS, J-TO-SP, SP-TO-J, PJ-TO-S, JS-TO-P) suffers from the same kind of asymmetry. The alternative of including all six functions in the principal schema would clutter this schema with needless redundancy.

Finally, a comment on the alleged advantage of a data model based on functions: namely that of making algorithms and data more interchangeable. The virtual data concept is very important, but its application is not at the logical level -- it is at a level that is oriented more to the strategy of storing and processing (i.e., a physical level). The relational model protects its users from concern with the underlying techniques used in materializing data.

Q2 Suppose a system-generated identifier is associated with each tuple of each relation and the identifier is unique at least within each relation. Would it not then be advantageous to replace every n-ary relation with domains D1,D2,...,Dn (say) by n functions that map the identifier domain into each of the ranges Di (i = 1,2,...,n) respectively?

A2 These functions are superior in symmetry to the set-valued functions discussed in A1 above. Nevertheless, they still suffer from two disadvantages compared with the n-ary relations they are intended to replace. First, while these tuple identifiers may be valuable instruments for search optimization within the system, they are not directly relevant to users' applications or interactions. Thus, users would need to be given languages which permit them to express their transactions without knowing of the existence of these identifiers. Such languages cannot therefore expose the proposed functions to their users. Hence, these languages must employ an entirely different data model from the one proposed. Second, such functions represent rather microscopic operands compared with n-ary

relations. If in a relational schema there are 50 relations with an average degree of 10, there would be 500 functions in the corresponding symmetric function schema. Now, consider how you would define in functional terms the counterpart of the natural join of two relations. The result of such a join is not, in general, a simple function of the proposed type. It is instead a collection of such functions, which suggests that a new data object (collection of functions) would have to be introduced.

Q3 How about replacing n-ary relations so that the totality of data is expressed in terms of binary relations?

A3 The two cases (asymmetric and symmetric) discussed in A1 and A2 above for functions apply also to binary relations. Functions are, after all, special kinds of binary relations; namely those that are not one-to-many. Introduction of these one-to-many binary relations does not affect the arguments cited above. The difficulties of representing relations in which there are three or more functionally independent (or nearly so) attributes remain undiminished.

With regard to the note to curious readers in Installment #3, a rough answer is that all three relations (BETWEEN, FLIGHT, SUPPLY) are many-to-many-to-many. However, there is a more incisive answer. As a clue, consider a ternary relation R(A,B,C) in which the only non-trivial functional dependency is that of C on the combination (A,B). Is not R just as hard to decompose as BETWEEN, FLIGHT, and SUPPLY?

In Installment #5 we shall deal with some questions concerning the semantics of relations.

Correspondence should be addressed to:

Dr. E. F. Codd
IBM Research Laboratory
K51-282
Monterey & Cottle Roads
San Jose, California 95193