

COMMENTS ON THE DEBATE:

DATA-STRUCTURE-SET VERSUS RELATIONAL

In the belief that it might be worthwhile for someone to attempt a summary of the "Great Debate," I have tried to set down my own observations and will pass them on to you for whatever they may be worth.

Suppose we consider that there will be three supported interfaces to future data base systems, and describe these suggestively as an End User Interface, an Application Programmer Interface, and a Systems Programmer Interface. Table 1 shows how these might very roughly be characterized. (Note: By systems programmer I am referring to the data base management system and not to the operating system.) There seems to be little argument that the End User Interface should have a basically relational view, or that some form of network view will be required for the Systems Programmer Interface. This is because the concept of independence from logical data structure more or less implies the relational view for the higher level, and the requirement to eliminate redundancy from the stored data more or less implies a network view for the lower level. I gather, then that the central thrust of the debate is over what form the Application Programmer Interface should take. This does seem to be a substantive issue, and not merely a "religious" argument.

Dr. Codd's position seems to be that the application programmer should be able to request data in relational terms and have it supplied by the data management facility into his work area, where he may then operate upon it in whatever fashion he wishes; or that he be able to call upon the data management facility to take data from his work area described in relational terms and store it into the data base. To the extent that the data base contains linkages and indices required to reduce redundancy and improve efficiency, it will be the responsibility of the data management facility to manage these, and they should be essentially invisible to the application programmer. In this way the viewpoint of the application programmer and the end user will be essentially the same, and communication between them will be easy. The task of the application programmer will be reduced and he will make fewer mistakes because he will not have to be concerned with how he acquires the data he wants to manipulate, but only with its specification.

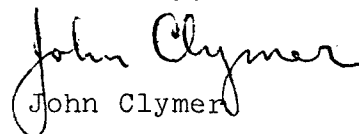
Mr. Bachman's position seems to be that the application programmer should be given the flexibility to chart his own course through the data, and that the system should not be doing a lot of things "below the interface" that he might not understand. By being aware of what the actual data structures are, the application programmer can thereby take advantage of them to write much more efficient programs. In this way, the viewpoint of the application programmer and the systems programmer will be similar and communication between them will be easy. The overhead of the data management facility will be reduced because it is only doing relatively simple things, and this may be an important consideration for routine tasks like payroll, and crucial when dealing with very large data bases.

These remarks represent only my own understanding of the debate, of course, and I would welcome further clarification.

If I may make one further observation, a fourth type of interface was mentioned and probably deserves some discussion. This is the interface to the Parametric User, i.e., the user who simply initiates a pre-determined function (possibly supplying one or more parameters to it). These parameterized functions can be pre-written in a host manipulative language, or can be pre-"canned" strings of statements in an end user language. There may be a question as to what extent such a parametric user must understand the data structures with which he is working (or to what extent the author of such a function must understand them), but I don't think this bears very directly on the debate.

I think the attached figure may provide some perspective. If we consider the end user interfacing to a normalization procedure which produces data requests in a normalized relational form, and consider these as being accepted by a data request evaluator which interfaces to the network language of the data base management system, then the combination of this data request evaluator and the data base management system may be termed a "data management facility." The viewpoint of the data base management system is a network view, and the viewpoint of the data management facility is a relational view. In these terms, the debate essentially revolves around the question of whether to place the Application Programmer Interface on the network side or the relational side of this data request evaluator.

Sincerely,


John Clymer

30 May 1974

TABLE 1

End User Interface

higher level
less procedural/"what" not "how"
data independent/independence of logical structure
less flexible
self-contained language
relational viewpoint

Application Programmer Interface

intermediate level
more procedural
data independent/independence of physical structure
more flexible
mixed language
? viewpoint

Systems Programmer Interface

lower level
procedural
data dependent
highly flexible
host language based
network viewpoint

