The Role of the Trade Press in Educating the Professional Community: A Case Study

by

C. J. Date

First written September 1993

Substantially revised May 2000

ABSTRACT

I was recently involved[*] in a series of exchanges with one of the leading IT industry trade publications. In what follows, I report on what happened and draw some conclusions from the experience.

----
[*] I.e., in 1993.

OVERVIEW

During the summer of 1993, a series of items appeared in *Computerworld* on the general topic of relational *vs*. object-oriented database management. The chronology of events was as follows.

1. First of all, at the request of Fabian Pascal (who was writing an article for inclusion in a forthcoming special "Object-Oriented Programming" feature issue of *Computerworld*), I wrote a short position paper entitled "Relational *vs*. Object-Oriented: Not an Either/Or Decision" (final draft dated March 12th, 1993). The idea was for that position paper to accompany Fabian's piece, thereby lending additional weight to the arguments he was

propounding in his own article. The intent behind my paper (as with most of the technical material I write) was primarily to *educate:* more specifically, to try to introduce some clear thinking into an area where most discussions still seemed to suffer from the "more heat than light" syndrome. The original draft of that paper is included in this report as *Exhibit A*.

2. *Computerworld* chose not to publish my paper as written, but instead published an edited and abbreviated version, with a different title ("A Fruitful Union"). While the editing -- which was done without any consultation with myself, I might add -- did not significantly change the message of the original draft, it did omit a few key points and it did alter the emphasis in some of what remained. (It also introduced some pretty slipshod writing, but let that pass.) That edited version, which appeared in the June 14th, 1993, issue of *Computerworld,* is included herein as *Exhibit B*.

3. Two weeks later (June 28th, 1993), *Computerworld's* technical editor Charles Babcock commented on the same subject in his regular *Computerworld* column, under the heading "Relational Backlash." His comments, it seemed to me, were definitely in the "more heat than light" category -- a fact I found both sad and rather annoying, given (a) my own position as stated in print, with supporting arguments, in his own newspaper only two weeks previously, and given also (b) the responsibility, as I see it, of writers in influential positions such as Babcock's to do their best to educate their readers instead of pandering to a taste (either real or perceived) for controversy. Babcock's article is reproduced as *Exhibit C*.

4. On June 29th, 1993, in response to Babcock's article, I wrote a letter to the editor of *Computerworld,* which was published in abbreviated form on July 12th, 1993. *Exhibit D* is the letter as originally written, *Exhibit E* is the version that was published.

5. The published version of the letter provoked a somewhat shrill response from a reader (Donald Burleson, *Computerworld,* August 2nd, 1993). I leave it to readers of this report to note how the omissions from my original article and from my letter enabled Burleson to make some of the comments he did. Burleson's letter (at least as published -- probably his original was edited too) is attached as *Exhibit F*.

6. I wrote a reply to Burleson's letter and sent it with a cover note to the editor of *Computerworld* (August 2nd, 1993). That letter was never published, nor even acknowledged. See *Exhibit G*.

7. Subsequently, another response to Babcock's column and to my July 12th letter was published (James Barnett, *Computerworld,* August 23rd, 1993). Comments similar to those under paragraph 5 above apply here also. See *Exhibit H*.

8. I wrote a reply to Barnett's letter and sent it with a cover note to the editor of *Computerworld* (August 24th, 1993). As with my August 2nd letter, that letter was neither published nor acknowledged. See *Exhibit I*.

THE SAGA CONTINUES

The day after I completed the first draft of this account, I opened up the most recent issue of *Computerworld* to find a follow-up column by Charles Babcock on the same topic, entitled "SQL *vs*. Objects" (see *Exhibit J*). Regrettably, though perhaps unsurprisingly, this new column also fell into the "more heat than light" category. In particular, it included, and commented on, a couple of quotes from my own article as published (i.e., as reproduced in *Exhibit B*). As might be expected, the comments were, on the whole, less than favorable. What's more, the first of the two passages quoted had been subtly changed in the published version (I had written "implemented," but the editor had changed it to "implements" -- compare *Exhibits A* and *B*), and that apparently trivial change made it possible for Babcock to take a very cheap shot at my original claims.

The rest of Babcock's column contained numerous errors and misleading statements. Detailed discussion of those errors *per se* would be beyond the scope of the present article; however, I have to say that I find it extraordinary that *Computerworld* should refuse to publish -- and publish *verbatim!* -- the carefully considered opinions of someone who has made a lengthy study of the subject, and at the same time see fit to give someone like Babcock the space to pontificate on a topic on which he clearly displays little real understanding.

CONCLUSIONS

What are we to conclude from all of the foregoing? Well, I'm only too well aware that any claim of shabby treatment at the hands of the media tends to smack of special pleading, and runs the risk of making the claimant appear ridiculous. And I'm also well aware that, in the case under discussion, the letters I wrote were at least as much an attempt at self-defense as they were an attempt to educate; a critic might therefore argue that I'm merely suffering from wounded pride. But reading through the exhibits in their entirety, it's hard to escape the impression that (in the case at hand, at least) *Computerworld* was more interested in controversy for its own sake than it was in the dissemination of truly useful information. Certainly it published more in the way of dissenting opinion -- ill-informed, unsubstantiated, and incorrect opinion at that -- than it did in the way of reasoned argument or material that was genuinely trying to educate the readership.

I'll leave it to readers of this report to judge whether this was an isolated incident or the norm. For my own part, I would like to continue to believe that there are still some trade publications that take their responsibilities seriously; my criticisms are reserved specifically for those that behave as *Computerworld* did in the case at hand. Be that as it may, here are some of the lessons I've learned from my experience in this particular case:

1. Technical articles and letters intended for publication in the trade press, no matter how carefully argued from a technical standpoint and no matter how carefully written, are likely to be altered for publication by editorial staff who don't understand either the substance or the merits of the material in question (and often have a tin ear to boot).

2. Many aspects of the IT field are not well understood by the professional community at large, and the quality of debate is thus often not very high either. Evidence for both of these claims can readily be found in the case under discussion. This state of affairs is hardly surprising, however (and moreover is unlikely to change), given the apparent lack of interest on the part of the trade press -- or certain portions thereof, at least -- in trying to improve the situation.

3. I will be very wary in the future of relying on the trade press for enlightenment on any technical subject.

This marks the end of the body of this report. Exhibits A-J follow immediately.

----- ----------- -----

EXHIBIT A

Original draft of "Relational vs. *Object-Oriented: Not an Either/Or Decision," by C. J. Date (March 12th, 1993):*

Where is database technology headed? Some pundits have predicted the imminent demise of relational, claiming that today's relational systems are just too simplistic for the complex databases we need in the 1990s, and have jumped with both feet on to the object-oriented (OO) bandwagon. "The world is much too complex to be represented in flat relational tables" is a typical claim heard from this camp. On the other hand, relational advocates have been defending their position stoutly, arguing the importance of relational's solid theoretical foundation, and pointing out -- with some justice -- that in certain respects OO technology represents a giant step backward.

Well, I have some good news: We can have our cake and eat it too! Relational *vs*. OO is not an either/or decision -- despite the fact that certain people have a vested interest in presenting it as if it were. The fact is, there are two technologies out there, relational and object-oriented, and we should be looking for ways to marry them together, instead of throwing mud at each other. The advantages and benefits of relational are well known; and object-oriented too has many good features, such as inheritance, that are missing from today's relational products. (It would be wrong not to point out in passing that inheritance in particular was proposed for relational systems as far back as 1976. As so often, there is a significant gap between what the technology is theoretically capable of and what the products actually deliver.)

Mind you, object-oriented has some very bad features too! This is not the place to go into details, but such matters as data integrity, optimization, views, and the catalog all represent areas where the object-oriented community has a good deal of work to do to catch up with relational technology. (Indeed, the foregoing might be too charitable. There are some serious questions as to whether such catching up is even feasible if we discard the solid relational foundation, as some OO people seem to be advocating.)

So when I talk of marrying the technologies together, what I mean is that we should be trying to extend relational systems to incorporate the GOOD features of OO. Obviously, I do not want to do this at the expense of having to incorporate the BAD features as well. And let me stress the point also that I am talking about a marriage of *technologies,* not of *products*. I am not pretending that a clean integration between OO product *X* and relational product *Y* is a simple matter, or even achievable, or even necessarily a good thing.

So how are we to meet this desirable goal? Well, the fundamental construct in OO systems is the **object class**, which is (in general) a user-defined, encapsulated data type of arbitrary internal complexity.[*] *Note:* I use the term "data type" here in the sense in which that term is understood in modern programming languages. In particular, I take it to imply that instances of the data type in question can be manipulated only by certain **operators** whose definitions are also provided by the user. I am NOT referring just to primitive, system-defined (i.e., builtin) data types like INTEGER and CHAR.

----

[*] *Note added in May 2000:* This sentence is a little misleading. An object class is just a data type, either user- *or system*-defined. User-defined is the more general case, of course, which is why the original sentence included that "in general" qualifier.

What about relational systems? Well, here the fundamental construct -- mostly not implemented, unfortunately, in today's relational products -- is the **domain**. And a domain is (in general) a user-defined, encapsulated data type of arbitrary internal complexity ... In other words, a domain and an object class are *the same thing!* In my opinion, therefore, domains are the key to achieving our "desirable goal." A relational system that implemented domains properly would be able to do all the things that OO advocates claim that OO systems can do and relational systems cannot. Thus, criticisms of relational from OO advocates may well be accurate if they are taken as criticisms of today's products, but they are NOT accurate if they are taken as criticisms of the potential of the technology.

To sum up: Relational vendors should do all in their power to extend their systems to include proper domain support. Indeed, an argument can be made that the whole reason we are getting into this somewhat nonproductive debate on the relative merits of OO and relational is precisely because the relational vendors have failed so far to support the relational model adequately. But this fact should not be seen as an argument for abandoning relational entirely. It would be a great shame to walk away from the experience gained from over 20 years of solid relational research and development.

EXHIBIT B

Edited version of the article shown in Exhibit A as published in Computerworld *("A Fruitful Union," by C. J. Date, June 14th, 1993):*

Where is database technology headed? Some pundits have predicted the imminent demise of relational databases. They claim today's relational systems are just too simplistic for the complex databases we need in the 1990s and have jumped with both feet onto the object-oriented bandwagon.

"The world is much too complex to be represented in flat relational tables" is a typical claim heard from this camp. On the other hand, relational advocates have been defending their position stoutly, arguing the importance of relational's solid theoretical foundation and pointing out that in certain respects, object-oriented technology represents a giant step backward.

Well, I have some good news: We can have our cake and eat it, too! The point is to marry the two technologies instead of throwing mud at each other.

When I talk of marrying the technologies, I mean we should try to extend relational systems to incorporate the good features of object orientation and shun the bad. Let me stress that I am talking about a marriage of technologies, not of products. I am not pretending a clean integration between object-oriented product *X* and relational product *Y* is a simple matter -- or even achievable or a good thing.

So how are we to meet this desirable goal? By looking for what the two have in common.

The fundamental construct in object-oriented systems is the object class, which is (in general) a user-defined, encapsulated data type of arbitrary internal complexity. (*Note:* I use the term data type here in the sense in which that term is understood in modern programming languages. In particular, it means only certain operators, whose definitions are provided by the users, can manipulate instances of the data type in question. I am not referring just to primitive, system-defined, built-in data types such as Integer and Char.)

In relational systems, the fundamental construct is the domain, which for the most part is not implemented in today's relational products. In general, a domain is a user-defined, encapsulated data type of arbitrary internal complexity -- i.e., a domain and an object class are the same.

In my opinion, therefore, domains are the key to achieving our "desirable goal." A relational system that implements domains properly would be able to do all the things that object-oriented advocates claim that object-oriented systems can do and relational systems cannot. Thus, criticisms of relational from object-oriented advocates may well be accurate if they are taken as criticisms of today's products; however, they are not accurate if they are taken as criticisms of the potential of the technology.

Relational vendors should do everything in their power to extend their systems to include proper domain support. Indeed, you can make an argument that the whole reason we are getting into this debate on the relative merits of object-oriented and relational is precisely because the relational vendors have failed so far to support the relational model adequately. But this fact shouldn't be an argument for abandoning relational entirely. It would be a great shame to walk away from the experience gained from more than 20 years of solid relational research and development.

EXHIBIT C

Charles Babcock's Commentary *column from* Computerworld *("Relational Backlash," June 28th, 1993):*

“You know that what used to be the younger generation isn't so young any more when its leaders start taking shots at the talent coming up behind them. That's what's happening now with some of the proponents of relational databases as they train their sights on object database management systems (ODBMS).

“This is strange because at one time, the hoary guardians of hierarchical and Codasyl systems said the same sort of things about relational. John Cullinane, president of the late Cullinet Software, used to tell me, "IBM may ship a lot of copies of DB2 but they're all sitting on the customers' shelves." *Right*. Cullinet was acquired by Computer Associates about a year later.

“Once again, you can hear the volume of disparaging comments beginning to pick up: "You don't need object-oriented systems. Relational can do everything they can do ... You can store unstructured data in relational tables ... Object-oriented results in a loss of data independence, not a gain." And so on.

“It is hard for relational advocates, having been on the leading edge for 10 to 12 years, to wake up and find that fashionable opinion has moved on to something else. The temptation is to tell the upstarts they don't know what they're talking about.

“At their worst, the relational defenders say ODBMSs represent a step backward. ODBMSs resemble the old Codasyl databases with their dependence on pointers to locate stored objects, but it is hard to see this as a vice if the systems then manage objects effectively.

“Object databases answered a real need for C++ and Smalltalk programmers who needed a place to store their persistent data. CAD and CAE users in particular sought to store objects, and object-oriented databases sprang up to serve that purpose.

“Whatever their deficiencies, ODBMSs succeed in dealing with objects as objects. They do not need to break them down or flatten them as relational systems do. There are a variety of methods used, but the chief one is to assign each object its own identifier and then use that identifier to locate the object intact.

“The nature of this system gives object databases a claim to speedier retrieval because there is no mathematical basis on which to do more sophisticated operations such as joins. Clearly relational systems would retain advantages in dealing with massive amounts of tabular data. But relational advocates are reluctant to give object databases their due.

“ODBMSs are built as object-handling systems capable of preserving the characteristics of the objects they store -- classes, inheritance and messaging.

“Because objects are a combination of data, processes and messages, it is difficult to restrict them to a few simple data types. To store an object, "you have to have the processes inside the database as well as the data," notes James Odell, chief methodologist at Inference Corp. and co-author with James Martin of *Principles of Object-Oriented Analysis and Design*.

“Relational systems can store objects, but to do so, they must break them down into components and store them in tables. In an analogy that originated with Esther Dyson, editor of the newsletter "Release 1.0," this is like driving your car home and then disassembling it to put it in the garage. It can always be reassembled again in the morning, but one eventually asks whether this is the most efficient way to park a car.

“Relational systems were designed to deal with a few data types within the confines of a strict logic. Object databases were designed to deal with the rich variety of data types in a few limited ways. Relational advocates can't wave a magic wand and make the difference go away.”

EXHIBIT D

There were many, many statements in Babcock's column that I would have liked to refute or at least comment on, but in my letter to Computerworld *I concentrated on what seemed to me to be the most important points. The original (June 29th, 1993) draft of that letter follows:*

Well, I guess I'm a "hoary guardian" of relational technology ... I refer to Charles Babcock's column "Relational Backlash" [CW, June 28]. But the analogy of disassembling your car to park it and reassembling it in the morning is just as hoary -- and what's more, it's WRONG. Relational technology does *not* necessarily require complex objects to be broken down into components to be stored in the database (see my article "A Fruitful Union" [CW, June 14]). Let me repeat the point: A relational system that supported **domains** properly would be able to all the things that OO systems can do. Moreover, it would still be a relational system and would enjoy all the usual relational advantages.

Also, the implication of Babcock's column that "OO is to relational as relational was to CODASYL" (paraphrased) is completely false and very misleading. Relational displaced CODASYL because it had a solid theoretical foundation and CODASYL did not -- and that foundation led in turn to solid practical benefits. OO does not have a comparable foundation. And yes, relational defenders do say that OO represents a step backward -- in some ways (e.g., the CODASYL flavor), though not in others (e.g., inheritance).

To paraphrase Babcock again: Relational technology was designed to deal with *arbitrary* data types (not "a few" data types) within the confines of a strict logic (this latter part is correct). What we want is for current relational systems to be extended to include the good features -- but not the bad features! -- of OO technology. OO advocates cannot wave a magic wand and make the bad features go away.

EXHIBIT E

This is the letter from Exhibit D as it actually appeared in print ("Reader

Backlash," by C. J. Date, July 12th, 1993):

Well, I guess I'm a "hoary guardian" of relational technology, according to Charles Babcock's column "Relational backlash" [CW, June 28]. But the analogy of disassembling your car to park it and reassembling it in the morning is just as hoary -- and what's more, it's *wrong*.

Relational technology does *not* necessarily require complex objects to be broken down into components to be stored in the database (see my article "A Fruitful Union," CW, June 14). A relational system that supported domains properly would be able to all the things that object-oriented systems can do. Moreover, it would still be a relational system and would enjoy all the usual relational advantages.

Also, the implication of Babcock's column that "object orientation is to relational as relational was to Codasyl" is completely false. Relational displaced Codasyl because it had a solid theoretical foundation and Codasyl did not -- and that foundation led in turn to solid practical benefits. Object orientation does not have a comparable foundation.

EXHIBIT F

Letter in Computerworld *from Donald Burleson, Rochester, NY ("Relational Redux [*sic*]," August 2nd, 1993):*

I was very surprised to see C. J. Date's naive response to Charles Babcock's "Relational backlash" [Letters to the editor, CW, July 12].

Mr. Babcock's analogy about disassembling a car every time it is driven is an excellent example of the overhead that relational systems impose on the computer industry. Relational databases have never been known as high-speed engines, and atomic data items that are shared by many logical objects do indeed require overhead.

It is ludicrous for Date to imply that proper "domain" support within a relational database would allow a relational database to do all the things an object-oriented database can do. I have yet to see any relational database that can fully support polymorphism or multiple inheritance. Most important, it is very difficult to "shoehorn" a relational database to support the encapsulation of behaviors.

Contrary to Date's assertion about the similarity of Codasyl [Conference of Data Systems Languages] to object technology, it is very clear he missed the point. Both models use pointers to link data items together, thereby reducing the overhead of reassembling the objects.

Object-oriented databases also share the concept of "currency" with Codasyl databases. Currency allows users to see where they are in the database, and the failure of the relational model to support currency is a major drawback.

Finally, Date alleges that the relational model is built on a sound theoretical foundation. Anyone who has read E. F. Codd's criteria for relational databases knows that attempts to apply mathematical rigor to the relational model fall apart in practice.

Relational databases sacrifice performance to remain flexible, and object technology databases sacrifice flexibility to gain performance. Much of this argument reminds me of the Codasyl bashing that was going on when the first commercial relational databases were introduced.

EXHIBIT G

Cover note to my letter to the editor of Computerworld *replying to Burleson's letter (August 2nd, 1993):*

The attached is submitted for publication in the *Computerworld* "letters to the editor" column. It is a reply to some comments on a previous letter. Since portions of my original letter were omitted from the published version and those omissions distorted my message somewhat, I would appreciate your publishing this reply in its entirety. Thank you.

The letter itself (which was not published) ran as follows:

It was with some reluctance that I ventured to comment (Letters to the Editor, CW July 12th) on Charles Babcock's "Relational Backlash" column of June 28th, since I strongly suspected that anything I said would be misunderstood. And I was right. Now, I certainly don't want to get into a lengthy debate in your columns, but Donald Burleson's letter (CW August 2nd) demands a response.

1. First of all, I don't think it's appropriate to characterize someone's opinions as "naive" or "ludicrous" when you manifestly either don't understand them and/or haven't bothered to read them properly.

2. I stand by my claim that a relational DBMS with proper domain support would be able to do all the (good) things an OODBMS can do. Burleson's remark that he has "yet to see any relational [DBMS] that can fully support polymorphism" in no way invalidates that claim, and betrays a lack of clear thinking. Ditto for the remarks about "disassembling a car every time it is driven."

3. I did not assert that CODASYL and OO technology were similar; I asserted that the suggestion that OO would replace relational just as relational replaced CODASYL was false.

4. The claim that relational DBMSs "sacrifice performance to remain flexible" is a hoary old canard. See, e.g., Tandem's NonStop SQL, Teradata's DBC/1012, etc., for some commercial counterexamples.

5. "Attempts to apply mathematical rigor to the relational model fall apart in practice"? Does Mr. Burleson really mean what he seems to be saying here? If so, I don't think he knows what the relational model is.

EXHIBIT H

Letter in Computerworld *from James R. Barnett, Deerfield, Ill. ("OOP Objections," August 23rd, 1993):*

“After reading Charles Babcock's article "Relational backlash" [CW, June 28] and C. J. Date's previous article "A fruitful union" [CW, June 14], about the merits of object *vs*. relational DBMS, it is my opinion that neither Mr. Date nor Mr. Babcock is completely correct.

“The bottom line is not whether one technology is better, but which technology best supports the requirements of the system being developed.

“I do agree with Mr. Date's statement that "relational vendors should do everything within their power to include proper domain support." However, his assertion that this would allow relational systems to do everything object-oriented DBMSs are capable of just isn't true. A domain and an object class are not the same!

“True domain support would still not allow the complex data types found in object-oriented systems. It would also not provide a mechanism for defining the valid methods for an object class. It would also not support object class hierarchies, which are the foundation of an object-oriented approach.

"Mr. Date's suggestion that relational is the better technology because it has a solid theoretical foundation and object oriented does not is a valid point. Object oriented may not have the same mathematical foundation; however, it's a more natural representation of the way people think and organize objects in the real world."

EXHIBIT I

Cover note to my letter to the editor of Computerworld *replying to Barnett's letter (August 24th, 1993):*

You saw fit not to print my recent letter responding to some remarks made by Donald Burleson (CW August 2nd) criticizing -- in a most ill-informed and *ad hominem* way -- my short article "A Fruitful Union" (CW June 14th). Please restore my faith in *Computerworld's* commitment to the dissemination of accurate, timely, and relevant technical commentary by publishing the attached letter in its entirety. Thank you.

The letter itself (which was not published) ran as follows:

With reference to the letter from James Barnett (Letters to the Editor, CW August 23rd): A domain and an object class *are* the same. True domain support *would* allow "the complex data types found in object-oriented systems." It *would* provide "a mechanism for defining the valid methods for an object class." It *would* support "object class hierarchies."

It is unfortunate that so few people seem to understand these simple facts. It is not however unexpected, given the low quality of much of the public debate in this area. Mr. Barnett's letter does nothing to improve the quality of that debate.

EXHIBIT J

Charles Babcock's Meta View *column from* Computerworld *("SQL* vs. *Objects," September 6th, 1993):*

"Object-oriented systems are gaining ground in some progressive IS shops, but their use is hampered by the fact that business data residing in relational database systems is not easily available to them. RDBMSs do not support the manipulation of objects, which means the data must be kept in two places or clumsily transferred in and out of the relational systems.

"For several years, relational vendors have been saying not to worry, object support is just around the corner. But as you listen to some relational authorities, their responses on this question start to sound suspicious. Instead of getting a time frame for object support, one tends to get a put-down of the young object-oriented DBMSs.

"In the June 14 issue of *Computerworld,* expert [*sic*] C. J. Date stated, "A relational system that implements domains properly would be able to do all the things that object-oriented advocates claim object-oriented systems can do ... ." This statement may be true as far as it goes, but it begs a very simple question: Which relational systems implement domains properly? The answer is, "none using SQL," which covers all the relational systems in commercial use.

"Date carried the argument a step further in a July 12 letter to the editor when he said: "Relational technology does not necessarily require complex objects to be broken down into components to be stored in the database."

"No, theories about the relational model *don't* require it, but SQL *does*. Which do you suppose database programmers are building their new systems with?

"Let's take a look at the SQL problem. In its initial implementation, SQL supported a handful of simple data types on which it could carry out its SELECT, JOIN and other operations. When the standard was expanded by the ANSI X3H2 committee in 1992, more complex data types were added, such as date/time, which could be made up of several pieces of data, all stored in a single field. But date/time is still a far cry from the unpredictable, user-defined mix of data types found in objects.

"The closest relational systems get to dealing with objects is the binary large object, also known as a Blob, which sort of sounds like the real thing but is actually a baby step in the right direction. Blobs tend to be images, long text strings, video or voice stored as a uniform data type in a kind of "binary bucket," in the words of Fred Carter, chief architect at Ingres.

"Storing and retrieving Blobs is not the same thing as full-fledged support of object-class libraries. The database management system can do little with a Blob except put it away and retrieve it. It usually takes an application to seize the data and reconstruct it into an image, etc.

"This limitation will give way as the ANSI X3H2 committee adds object support to the next version of SQL. SQL III may be published in 1995, "but more likely in 1996,"[*] says Jim Melton, a Digital database architect who serves as editor for the committee."

----

[*] *Note added in May 2000:* In fact it was 1999, and late 1999 at that -- and it would have been much later still, if the decision to publish had been based on technical considerations rather than political ones. From a technical point of view, SQL:1999 was, and still is, very seriously flawed. (This is not just my own opinion. Indeed, the SQL committee was *already* working on a major Technical Corrigendum at the very time the standard was being ratified! That Corrigendum has yet to appear, so whether it will address any of the really fundamental problems is unknown; my own guess is that it probably won't.)

"Oracle is playing an active role before X3H2 and is committed to including object support in Oracle8. Company spokesmen estimate Oracle8 will be available in early 1995. Informix, a pioneer in the field, also won't wait for the standard to be published before it adds object management features.

"But all of this remains somewhat up in the stratosphere. No one knows for sure when any vendor will have object support or how extensive that support will be. And the ANSI committee must act if support is to be uniform. ANSI panels have been known to fall behind their timetables on less complicated matters than this.

"To my mind, the interest in object-oriented programming has caught the relational vendors somewhat unawares. Their systems cannot be replaced willy-nilly by object-oriented database systems, and I do not know of any users contemplating doing so. But then again, some users will have to wait two to three years before an ingredient they need is added to their systems."

----- .... -----

POSTSCRIPT

To say it again, the exchanges described above took place in 1993, but I believe the overall message is just as relevant now as it was then. Note in particular that -- as I hope you will agree -- subsequent events in the IT field have tended to support the point of view I was trying to express in the article I originally wrote (*Exhibit A*). To summarize the current situation:

- Pure object-oriented DBMSs are now mostly seen for what they really were all along: *viz.,* as "DBMSs" that are *specific to some particular application area,* not general-purpose ones (and thus not true DBMSs, as that term is usually understood in the IT community, at all). As such, they might have a useful role to play in certain circumstances, but -- to repeat -- thay aren't true general-purpose DBMSs, and they represent something of a niche market. Certainly they'll never replace relational DBMSs.

- By contrast, "object/relational" DBMSs *are* true general-purpose DBMSs that do provide both relational and object capabilities. As I was saying all along, we really can have our cake and eat it too! And as Hugh Darwen and I have written elsewhere,* the idea of integrating relational and object technologies in such a manner is not just another fad, soon to be replaced by some other briefly fashionable idea. *Au contraire,* we believe an object/relational system is in everyone's future.

----

* C. J. Date and Hugh Darwen: *Foundation for Future Database Systems: The Third Manifesto* (2nd edition). Reading, Mass.: Addison-Wesley (2000).

- Of course, I feel bound to add that (to quote from that same book by Hugh Darwen and myself again), a true "object/relational" DBMS would be nothing more nor less than a true *relational* DBMS -- which is to say, a DBMS that supports the relational model, with all that such support entails (in particular, it entails proper *domain* support). The trouble is, the term "relational DBMS" has effectively been usurped by systems that are *SQL* DBMSs merely, so we need a new term ... Hence the "object/relational" terminology.

- Finally, it does need to be said that, unfortunately, the vendors are (once again) getting it wrong! I refer to the fact that every object/relational product on the market -- at least, every one I'm aware of -- is committing at least one of **The Two Great Blunders**. For details, I refer you once again to the book by Hugh Darwen and myself already mentioned a couple of times above; here I just want to raise the question: *Why* are the vendors getting it wrong? The answer, it seems to me, is because the relational model is so widely misunderstood. Why is it so widely misunderstood? Because of the lack of good education. Why is that good education lacking? ... *Now go back to the beginning of this report (starting with the title!) and read it all over again*.

***** End *** End *** End *****