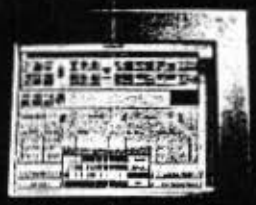
Alpha 5 for Windows • Client/Server Security: Are You Prepared?

DATABASE PROGRAMMING & DESIGN

InSync's Passport to OO



SEPTEMBER 1995 VOLUME 8 • NUMBER 9

Large Client/Server Applications

How to Build from Strength

Most Frequently Asked Questions for Data Modelers

Database Fingerprinting

A Solution for Configuration Management

A MILLER FREEMAN PUBLICATION

$3.95 ($4.95 Canadian)

09>

0 70989 35537 8

DATABASE

PROGRAMMING & DESIGN

VOL. 8 NO. 9 SEPTEMBER 1995

FEATURES

COLUMNS

DEPARTMENTS

COVER ART Mike Fisher

DATABASE PROGRAMMING & DESIGN [illegible] is published monthly by Miller Freeman Inc., [illegible] ... All material published in DATABASE PROGRAMMING & DESIGN is [illegible] ... University Microfilms International, 300 N. Zeeb Rd., Ann Arbor, MI 48106 [illegible]

C. J. DATE, HUGH DARWEN, AND DAVID MCGOVERAN

Nothing to Do with the Case

The debate over the use of many-valued logic (MVL) in relational databases has been scorching a path through Database Programming & Design's pages for several years now. The latest round began with C. J. Date's "According to Date" columns of December 1992 through March 1993 (as well as his writings published elsewhere). The columns prompted a searing reply from none other than Dr. E. F. Codd, father of the relational model. Codd, who supports some forms of MVL, countered Date's criticisms and clarified his own views on the topic. Codd's commentary, along with rebuttals from Date, were featured as a "debate" in our magazine two years ago ("Much Ado About Nothing," 6[10], October 1993).

For two months, all was quiet. Then, beginning in the December 1993 issue and carrying on through much of 1994, Database Programming & Design carried an important series of articles by David McGoveran. McGoveran's "Nothing from Nothing" series described, first, the logical problems with MVL; second, why relational DBMSs do not support the relational model's underlying logic; and finally, how designers can avoid having to use MVL. Date and McGoveran took these ideas further, teaming up to describe a "New Database Design Principle" in a series of articles in the summer of 1994.

By 1995, it was about time for a counterpunch: This punch was provided by Tom Johnston's "MVL: Case Open" and "The Case for MVL" (February, 8[2], and March, 8[3] issues). Johnston, using extensive examples regarding underlying logic and relational database design, defended Codd's views and strongly countered those of McGoveran, Date, and Hugh Darwen.

Given the extent of Johnston's articles, we felt that Date deserved a chance to defend his (and his cohorts') views, and offer a critique of Johnston's articles. In the name of fairness, we will provide Johnston with an opportunity to rebut this article (his comments will appear in our November issue). With that, we'll all shake hands and end this chapter of The Great MVL Debate.

That is, once our readers have their say: We have already published many excellent letters on this topic in our Access Path section. We hope you have found the discussions about MVL enlightening and helpful to your efforts in solving this difficult relational database design dilemma: that is, how do we represent unknown or missing information in the database? —*Editor*

Last February and March, Tom Johnston presented his case for supporting many-valued logic in relational databases. This month C. J. Date and friends forcefully rebut those views

TOM JOHNSTON'S TWO-part article strongly criticizes the published views of myself and, to a lesser extent, Hugh Darwen and David McGoveran on the subject of many- (or multi-) valued logic and its suitability as a basis for dealing with missing information in databases.[1] Since it is my writings that seem to be Johnston's primary target, it seems appropriate that I should be the primary author of this response. However, I will use the first person plural when making statements that reflect the joint position of the three of us (almost always, in other words!). Also, Hugh and David have reviewed this response in its entirety and both have some comments of their own to add at the end.

OVERVIEW

The problem of how best to deal with missing information in databases has long been a thorny one. Johnston's article does not move us any closer to solving that problem. In particular, it totally ignores most of our documented objections to the idea of using three-valued logic (3VL) as a basis for any such solution. Those objections are discussed in detail elsewhere;[2-6] there's no point in repeating the specifics here, but I will at least give a list of some of our more serious objections later in this response. We remain convinced that the database field must look elsewhere for possible solutions to the missing information problem.

Johnston makes three claims in support of his contention that DBMSs should be based on many-valued logic (MVL) instead of two-valued logic (2VL):[7]

1. 2VL is just as "flawed" with problems of interpretation as MVL is.
2. "Basic" logic sacrifices a great deal of expressive power.
3. Many of "Date's objections" to MVL are inconclusive.

Note: While I do understand Johnston's reasons for using such expressions as "Date's objections" (and "Date's arguments," and "Date's recommendations," and so on and so forth), I have to say that I'm not very comfortable with them; they give the reader the impression that I'm the only one who believes in those objections and arguments and all the rest. On the contrary: The positions I've articulated in my various publications on this subject are supported quite widely in the database community, and with very good reason.

I now proceed to examine Johnston's three claims. *Note:* In what follows, all otherwise unattributed quotations are from Johnston's article.[1]

2VL FLAWS VS. 3VL FLAWS

I've argued elsewhere that, in 3VL, NOT is not *not*—meaning that the NOT of 3VL is not the same as the *not* of ordinary English.[4] This claim is undeniably true and has demonstrably led to errors, both in vendor DBMS implementations and in user queries—even (in the latter case) when the user in question was familiar with 3VL.[8] In Part I of his article, Johnston devotes a very great deal of space to arguing what we all know—namely, that even in ordinary *two*-valued logic, AND is not *and*, OR is not *or*, and IMPLIES is not *implies*. He concludes that:

> "To paraphrase Quine, . . . AND is merely the truth-functional distillate of *and* [and similarly for OR and IMPLIES, presumably] . . . No significant difference exists between 2VL and MVL in point of counterintuitivities of interpretation . . . 2VL is just as logically flawed as any MVL . . . MVL is actually *less* misleading than 2VL, because with MVL, the counterintuitivities are so obvious . . . Just because [those counterintuitivities] are so startling, they are less likely to trip us up than [the] subtler counterintuitivities [of 2VL]."

Now, we can surely all agree that the operators (or connectives) AND, OR, and IMPLIES are open to being misunderstood, even in 2VL; they are indeed "distillates" of their natural language counterparts. (It is worth pointing out, however, that the reason for this state of affairs is that the meanings of the logical operators are required to be *context-free*, as David McGoveran will explain later in his response.) Nevertheless, I would contend precisely that these operators are distillates of their natural language counterparts in a way that the NOT operator of 3VL is not! (If you see what I mean.) In 3VL, moreover, AND, OR, and IMPLIES also seem to depart further from their natural language counterparts. As already mentioned earlier, it is undeniable that DBMS *implementers* (not just users) have made mistakes over the 3VL operators that they didn't make with the 2VL versions. Johnston's claim that MVL is "less likely to trip us up" than 2VL isn't supported by the evidence.

Furthermore, if the 2VL operators were as much a problem as Johnston suggests, the whole business of computing would never have gotten off the ground in the first place! 2VL is relevant to the entire computing field, not just to that particular subfield we happen to call database management. (I'm not saying that no mistakes have ever been made in 2VL, but those mistakes haven't been nearly as pervasive as those in 3VL have been.) To put this point another way: I'm quite comfortable with the idea of having to teach "naive end users" how to use 2VL correctly; I'm not at all comfortable with the idea of having to teach them how to use 3VL correctly (and I speak from experience here).

As a further conclusion to his first argument, Johnston also claims to "have demonstrated that 2VL is just as logically flawed as any MVL," and hence that MVL is "just as secure a mathematical foundation for relational theory as 2VL." Here I'd just like to point out that even if we were to accept these claims—which we don't—of course it wouldn't follow that MVL is a *good* foundation, nor that MVL is better than 2VL for such purposes.[9]

Incidentally, Johnston repeatedly accuses me of, and takes me to task for, referring to the difficulties with 3VL (and MVL) as *logical flaws*: "If . . . Date is referring to MVL as a system of *mathematical* logic, then neither Date, Darwen, nor McGoveran have pointed out a single logical flaw in MVL." Actually I can find only one remark in all my writings on this subject that could reasonably be construed as suggesting that I think that 3VL is logically flawed. In my published debate on the subject with E. F. Codd, I said the following:[10]

"It seems to me that there is all the difference in the world between:

- Building a system—that is, one based on 3VL—in which we *know* errors will occur, because the system has logical flaws in it, and
- Building a system that is at least logically correct but is open to misuse. *Any* system is open to misuse. That's why we have to have discipline."

This remark was perhaps a little hasty; certainly I realize that it would be possible to define a 3VL that is logically self-consistent, and I never meant to suggest otherwise. What I did mean was that, in a system based on such a 3VL, certain conclusions will follow that are "logically incorrect" *in the real world*. The most obvious example is the well-known fact that the expression *p* OR NOT *p* is not a tautology (that is, it's not identically true) in 3VL. Other examples abound.

(As an aside—and to head another argument off at the pass—I should perhaps point out that I certainly have claimed that *SQL's attempt* to *implement 3VL* suffers from logical flaws.[11] But this claim is very different from the claim that 3VL itself is logically flawed, and I still stand by it.)

STAYING WITH BASIC LOGIC

Johnston uses the term *basic logic* to refer to "the logic on which the original version of relational theory was based"—namely, two-valued propositional (and first-order predicate) logic. And he claims that "Date apparently wants to remain within the confines not just of 2VL, but of [basic logic]"—although he does also go on to mention that he doesn't "know whether [Date] has ever said this" but has "never seen an example [in which Date] went

beyond these confines." This disclaimer seems a little perfunctory, however, or even disingenuous, since Johnston then devotes virtually the whole of Part II of his article to criticizing my imputed position that DBMSs should "remain within the confines . . . of basic logic."

In response to this broadside, I observe first that Johnston is quite right to say that "basic logic" is the logic on which relational theory was originally based. Thus, it would be more accurate to say that "remaining within the confines" of that logic was *Codd's* original position, rather than ours (or mine).

Be that as it may, I feel bound to say that it's very distressing to be criticized so roundly for something I didn't do. I never intended to rule out the possibility of exploiting such disciplines as modal logic and the rest, and I don't believe any of my writings suggest otherwise. However, I do believe strongly that any attempt to incorporate such concepts into our DBMSs should be approached with considerable circumspection; the price for making mistakes in such matters is far too high. In other words, I subscribe to *The Principle of Cautious Design* (and so do my colleagues).[12]

Much more to the point: All this discussion of modal logic and so forth is a *complete red herring*! It has nothing whatsoever to do with the matter at hand—which is, to repeat, the suitability or otherwise of MVL as a basis for dealing with missing information in databases. Most of our publications in this field have dealt, very specifically, with the clear *unsuitability* of the brand of 3VL (and 4VL) advocated by Codd—and supported, in a flawed fashion, by SQL—for this purpose.[10, 14, 15] We certainly don't see anything in Johnston's article as a valid defense for this particular brand of 3VL (or 4VL).

Incidentally, Johnston says at one point that he "will venture a guess . . . as to what Date and company's philosophical principle might be." There's no need to guess! Our principle is well documented and is a matter of public record; it's *The Principle of Cautious Design* already mentioned.[12] In other words, Johnston is quite right in suggesting that our position on this whole topic is "conservative" rather than "liberal." Perhaps the point needs spelling out that we're not talking about politics here, we're talking about the best way to design and implement software systems—systems of a rather fundamental nature, too, systems in which errors of design can have calamitous consequences. Liberalism might be highly laudable in

the political arena, but it doesn't follow that it's a good position to adopt in the arena under discussion.

INCONCLUSIVE OBJECTIONS?

To repeat from Johnston's article: "Date's objections to MVL are not as conclusive as they may appear at first glance." Since Johnston does not even address most of those objections, the most charitable thing that I can say about this claim is that it remains unproven. Here for the reader's benefit is a list (it is not annotated, for reasons of space) of some of those unaddressed objections. For further discussion, see our many previous publications on this subject.[2-6]

- Mathematically speaking, a "relation" that "contains a null" isn't a relation (and so relation theory doesn't apply).
- A "relation" that "contains a null" violates Codd's own Information Principle.
- If TABLE_DEE represents *true* and TABLE_DUM represents *false*, what represents *unknown*?
- What's the relation predicate for a "relation" that permits nulls in one or more of its columns?
- The fundamental theorem of normalization breaks down in 3VL.
- What's the justification for treating nulls as duplicates of one another for the purposes of union and projection but not for the purposes of restriction and join? (In other words, are two nulls equal or aren't they?)
- Even if such a justification can be found, why is it necessary for the relational algebra to suffer such complication?
- Expressions that evaluate to *true* in 3VL are not necessarily true in the real world (in other words, "3VL doesn't match reality").

And so on (this list isn't exhaustive).

DEFAULT VALUES VS. MVL

Here I really do object, vehemently. Johnston misrepresents, or misconstrues, the default values approach and then attacks his own misconstruction. "When we say that Jones doesn't have an hourly rate, we don't mean that he has some special hourly rate—the $0.00 rate, for example." I agree, we don't. What we do do is this: We say that the domain for the HOURLY_RATE column is—let's say—X, where X consists of all possible valid hourly rates *plus* the default value "N/A." The system certainly is capable of recognizing this default value as meaning "not applicable" and acting accordingly (and indeed should do so).

There are so many points I want to make regarding this particular issue that I think it best to save them for a separate article—perhaps a column in my regular column in *Database Programming & Design*. For now, I'll just have to refer readers to my own published description of the default values approach.[13]

MISCELLANEOUS COMMENTS

In this section I want to respond to a number of miscellaneous claims and observations in Johnston's article.

- "One of [Date's] arguments—the *infinite regress of truth-values* argument—does not apply to many standard versions of MVL."

I never claimed that the MVL approach (more accurately, the idea of using "nulls" instead of values to represent the fact that information is missing) implied an infinite regress of truth values; rather, I claimed that it implied "an infinite regress" of *nulls*.[4] These two claims aren't the same thing at all! In fact, Johnston's error here might well be called a howler. Although it's not irrelevant to point out that many other writers, in-

eluding both Codd himself and the authors of the draft "SQL3" proposals, have also fallen into this same trap of equating nulls and truth values at one time or another.[15,16]

• "When *p* represents any of several different kinds of self-referential statement, . . . tautologies [such as *p* OR NOT *p*] become contradictions."

Actually, they become *paradoxes*—that is, statements that are neither *true* nor *false*. (A contradiction is a statement that is identically *false*; it's the opposite of a tautology, which is a statement that is identically *true*.)[17]

• "It should be an object lesson to those who dream of relational databases as contemporary versions of such self-evident systems that Frege actually got it wrong!"

In connection with this remark, Johnston imputes to us the claim that we would like to realize an "ancient [but unachievable] philosophical dream." We've never made such a claim; indeed, we've shown that we know that the "philosophical dream" in question can't be realized.[18]

• The "proof" that basic logic is nonmonotonic is specious. What it boils down to is the following:

1. Queries produce results from database states.

2. "Adding an axiom" means changing the database state (it's just a fancy way of saying "update the database").

3. Changing the database state will certainly change the answers to some queries (it would be pretty useless if it didn't!).

• The penguin example is specious too, for similar reasons. (I can't resist the temptation to refer to it as a *red penguin*; perhaps it subsists on red herrings.)

• "Standard two-valued, first-order predicate logic is *not* truth-valued!"

This claim is correct as stated, because "standard two-valued, first-order predicate logic" permits infinite sets and quantification over them. In databases, however, we deal only with finite sets, and Johnston's point is irrelevant.

• "It may surprise some to realize that MVL is consistent with bivalence."

It certainly surprises me! (and it's a very strange remark to tuck away in the references, if it's really as significant as it seems to be). "Bivalent" means two valued; the "MV" in MVL means, well, many valued, where "many" means greater than two.

Perhaps Johnston is referring to a system in which every truth value except *true* itself is (at some point) effectively converted to *false*. SQL's 3VL might be thought of as behaving in this way, since

(for example) SQL queries retrieve only those rows for which the defining condition evaluates to *true*, not those for which it evaluates to *false* or *unknown*. In other words, SQL effectively converts *unknown* to *false*—*but it does this only at the outermost level of the query*. In effect, SQL uses 3VL internally, but switches back to 2VL "at the last moment." If this is all that Johnston's remark means, then his point is irrelevant, and our criticisms of 3VL are still germane.

• "I think it is clear that the supervaluational approach is not at all unwieldy."

This conclusion is far from clear to me, and I suspect it isn't valid. Is it always possible to recognize tautologies and contradictions? Even if it is possible, how feasible is it to do so? (In this connection, it might be worth mentioning that—as a correspondent, Ceuan Clement-Davies from Frankfurt, pointed out to me in a letter following the publication of my debate with Codd[10]—such a proposal seems to mean that the DBMS "should use *two-valued* logic to detect tautologies, and *three-valued* logic for everything else. This would be a curious mixture.")

• "That RDBMSs could eliminate these [join] costs, or at least keep them to a manageable level, is a misconception put forward by vendors and, I regret to say, by Date himself."

Here I cry *FOUL*! This comment is gratuitous at best (it has nothing to do with our complaints regarding 3VL, nor with Johnston's objections to those complaints); moreover, the part that refers to me is untrue. I've criticized the vendors for years, not least in the columns of this magazine, for delivering inadequate levels of independence between the base and stored table levels. (Such lack of independence is the crux of the issue here; if the vendors provided a greater degree of such independence, then join costs could indeed be "eliminated or kept to a manageable level.") And I cry *FOUL* again, for much the same reason, in connection with the second paragraph of annotation to Johnston's reference in Part II of his article.[8]

CONCLUSION

Johnson's article concludes: "By restricting ourselves to basic logic, we spurn a logic that wears its difficulties on its sleeve for one that hides its difficulties up its sleeve" (and so we should indeed opt for MVL instead of a 2VL). We find this conclusion prettily phrased but unsupported by the arguments that precede it. Our own conclusion, by contrast, is that we find absolutely no reason in Johnston's article for modifying our position.

HUGH DARWEN ADDS:

Chris Date has already said nearly everything I wanted to say and more, and I agree with it all. Above all, Johnston has not risen to the biggest challenge facing those who would like to promote MVL as a foundation for database systems: He has not given us a replacement for, or even a suggestion for a replacement for, the Relational Model of Data (including abstract machines to replace the relational calculus and algebra, and a database design methodology to replace the relational methodology based on dependency analysis, especially functional dependency analysis). He writes, in his conclusion to Part I of his article, "I believe that there is not adequate reason to enshrine basic logic as *the* logic for relational database access languages." But relations just do "enshrine basic logic," by *definition*, so this statement makes about as much sense as would "I believe there is not adequate reason to enshrine the basic operators *plus*, *minus*, *times*, and *divide* as *the* set of operators for doing arithmetic." If Johnston has in mind some alternative structure, analogous in some sense to the relation but enshrining MVL, then could he please tell us about it?

Does he perchance have in mind something close to what SQL pretends (in both senses of that word) to support? Take the example of the EMPLOYEE_EARNINGS relation in which certain symbols appearing in the BONUS column are to be severally interpreted as "a bonus of $1,000," "a bonus of $2,000," "some bonus," "no bonus," "a bonus of $0," "some level-3-security-classified bonus," and so on. I'm a simple soul. I cannot see any way of preventing the DBA from designing the database in such a manner—and I would have to change hats before I could comment on the advisability of doing such a thing. I would just observe that the data type (domain) of that BONUS column is *not* MONEY (although I can imagine a function that would easily map a certain proper subset

of that domain to values in the MONEY domain). I do understand that something called "modal logic" *might* be able to help us handle such domains in a general way, but it isn't clear to me that such a mechanism would necessitate a departure from 2VL.

Johnston rolls out the old problems of interpretation that arise with the logical operators *and*, *or*, and *implies*. We have never denied that these problems exist, but they aren't remedied by MVL. Besides, logical *and* really does mean one of the things human "and" means, and logical *or* really does mean one of the things human "or" means, and it's not difficult to explain to human beings which of the meanings is the applicable one in both cases. Logical *implies* means everything that human "implies" means; it's just that humans use only half of what the logical operator means, and are sometimes a little puzzled as to why logicians bother with the other half. (I, an amateur in the field of logic, have sometimes been surprised to find myself defending the logician's position in hot debate with *mathematicians*!) These "counterintuitivities" should be put into perspective against those that arise with 3VL.

By the way, I don't believe I've ever used the phrase "counterintuitivities of interpretation," which, personally, I can't make sense of. I *did* use the phrase "difficulty of interpretation," arising from the "traps and counterintuitivities" in MVL, which does make reasonable sense (to me). And I *might* have written, if only I had been bold enough, "*errors* of interpretation," but then our gainsayers would probably have accused me of below-the-belt rhetoric. But our gainsayers are mostly silent ones, in public, so I am at least very grateful to Tom Johnston for his efforts in reopening the debate.

DAVID McGOVERAN ADDS:

While I am sure Tom Johnston is well-intentioned, I think he has missed numerous important points and misinterpreted many statements from my series of articles.[5] Comments 1-7 below apply to Part I of Johnston's article; comments 8-12 refer to Part II.

1. Johnston's use of the term "interpretation" is very different from mine. I have consistently used this term in the Tarskian sense (except where I have explicitly referred to "semantic interpretation"). My concern has always been with providing a consistent formal interpretation along with a semantic interpretation having some relevance to database work. Users of logic must understand that arbitrary assignments of meaning to logical operators and truth valuation symbols can completely destroy the value of a logical system; some assignments work and others do not. I have yet to see an explicit 3VL (or 4VL, 5VL, . . .) that has the required properties, and indeed argued in my earlier articles that such a logic was impossible.[6]

2. Throughout his articles, Johnston confuses the need for a computational apparatus with the issue of the ambiguities of the English language. He proposes the use of such logics as *quantification over propositions, modal logic, second-order predicate logic, relevance logic, epistemic logic, deontic logic*, and *nonmonotonic logic* for a DBMS. But such logics are, in general, either incomplete or inconsistent, have esoteric interpretations (and are therefore very hard to use at all), and/or may be completely noncomputational. They cannot be used for automated query optimization and therefore do not provide logical data independence! We can reasonably assert that they are inconsistent when lumped together in a single logical system.

3. My usage of *true* and *false* differs from Johnston's, particularly in that I do not equate propositions and statements (and certainly not utterances), nor do I use the terms "proposition" and "statement" interchangeably.

4. The issues raised by Johnston regarding the interpretation of English statements into formulas of logic have no relevance to the issues at hand. English is context-sensitive and, as such, beautifully ambiguous. It is precisely this ambiguity that allows us to carry on conversations, as I learned as a graduate student from Jim McCawley (see Johnston's reference in Part II of his article).[6] But in any particular context, English statements are translatable into logical formulas. Even future contingents such as "It will rain tomorrow" can be formulated logically; for example, we often understand this statement to mean "So and so claims it will rain tomorrow," but we rarely construe it to mean something so simplistic and Aristotelian as "Rain happens tomorrow"! The first of these two construings is classifiable as *true* or *false*, while the second can only be classified as some kind of nonsense. Databases are holders of assertions made by reasonable people; each row inserted in the database is *an assertion of* a fact, not an absolute fact. We can't do anything about this state of affairs, since knowledge is always somewhat relative. You can never *logically prove* that you know anything.

5. Every elementary schoolchild studying simple set theory learns that AND does not mean English "and then" and that INCLUSIVE OR is to be distinguished from EXCLUSIVE OR. Yet Johnston seems to think these are subtle issues. Logical OR, AND, and IMPLIES are not simply distillates of their natural language counterparts, since logical connectives had to be made context-free. This freeing of the connectives from context has always been a major aspect of 2VL's evolution. By contrast, I have never seen any evidence that the development of 3VL was due to an attempt to make the meanings of the connectives more "natural language–like"; rather, it seems always to have been a matter of circumventing the restriction that connectives apply only to propositions and well-formed formulas!

6. Johnston wrongly asserts that P0 IMPLIES Pi somehow means that any false proposition P0 implies the truth of all disjunct propositions Pi. Instead, it is the entire proposition "P0 IMPLIES (P1 OR ... OR Pn)"—and not the disjunction P1 OR ... OR Pn, nor any individual Pi alone—that is *true*. Perhaps Johnston is confusing this issue with the more common concern of being able to prove anything from a contradiction. The proper understanding of IMPLIES would be "accepting the truth of a false proposition implies the truth of all other propositions." Expressed colloquially: "If you'll accept a falsehood, you'll accept anything."

7. The exercise at the end of Part I of Johnston's article consists solely of problems with ambiguities of English statements presented without a clarifying context. The exercise has nothing to do with logic!

8. The reference to Russell's Paradox—"Is the set of all sets a member of itself?"—

and Frege is gratuitous and irrelevant.

9. Johnston's definition of the Closed World Assumption is correct for systems involving quantifiers over predicate variables having domains that are nonfinite sets, but is not correct if all predicate variable domains are finite. In this latter case, it is crucial to define the finite domain of discourse for all predicate variables. The Closed World Assumption then means that if P is not an axiom nor a theorem of the system, either it is outside the domain of discourse or we may add NOT P as an axiom. Johnston's definition leads to logical systems that are infinitely extensible with new axioms.

10. Johnston gives an argument against truth functionality, asserting that "basic logic" is not even truth functional. Unfortunately, the implied definition of truth functional is simply wrong.

11. Contrary to Johnston's implication, standard 2VL first-order predicate logic is truth functional as long as quantification is over predicate variables defined on finite domains (which entails a special version of the Closed World Assumption). Regarding Johnston's alternative supervaluational approach, I would love to see an algorithm for identifying tautologies and contradictions—or a proof that any such algorithm either is or is not NP-complete.

12. Claims that RDBMSs cannot optimize for the join I/O problem are ludicrous. I would be pleased to consult for any vendor interested and would even offer a discounted rate if they would implement the algorithm in question!

There are so many other errors in Johnston's article that I would have to spend a great deal of time correcting and explaining all of them. Some of them represent failures to understand myself, Date, and Darwen; others are due to incorrect application of formal logic; still others are simply false assertions about the history of logic; others are due to a lack of understanding of what McCawley referred to as "linguistic logic" and natural languages.

It is unclear to me how best to help readers understand the negative impact of such articles and avoid the dangerous conclusions they draw. All I can do is suggest that, before you support esoteric bases for your DBMS, you should be sure that your company can afford to risk the validity of your applications and the integrity of your data. At least with 2VL, we understand what needs to be taught, we understand what needs to be included in DBMS query languages and optimizers, and we understand how to avoid any mechanical (that is, automatic or computational) errors. Thus, we do at least have some hope of identifying other mistakes as human errors and striving for better use of the products. The whole point of computerization is not to automate errors, but to automate correct (possibly complex) procedures—thereby leaving to people the noncomputational tasks involving judgment, intuitive recognition, and creativity—and (hopefully) more time for these enjoyable activities.

NOTES AND REFERENCES

1. Johnston, T. "MVL: Case Open." *Database Programming & Design*, 8(2): 52-63, February 1995; "The Case for MVL." *Database Programming & Design*, 8(3): 54-68, March 1995.

2. Darwen, H. "Into the Unknown." In C. J. Date, *Relational Database Writings 1985-1989*. Addison-Wesley, 1990.

3. Date, C. J. "Null Values in Database Management." *Relational Database: Selected Writings*. Addison-Wesley, 1986.

4. Date, C. J. "NOT Is Not "Not"! (Notes on Three-Valued Logic and Related Matters)." *Relational Database Writings 1985-1989*. Addison-Wesley, 1990.

5. Date, C. J. "Three-Valued Logic and the Real World"; "Oh No Not Nulls Again." Both in C. J. Date and Hugh Darwen, *Relational Database Writings 1989-1991*. Addison-Wesley, 1992.

6. McGoveran, D. "Nothing from Nothing" (in four parts). *Database Programming & Design*, 6(13): 33-41, December 1993; 7(1): 54-61, January 1994; 7(2): 42-48, February 1994; 7(3): 54-63, March 1994.

7. This contention on Johnston's part notwithstanding, our published papers on this subject have been primarily concerned with 3VL specifically, rather than with MVL in general. (David McGoveran's paper is an exception.[6]) However, most of our criticisms of 3VL apply to MVL also, often with even more force. Following Johnston, however, I will talk in terms of MVL in this response, except where I want to say something regarding 3VL specifically.

Incidentally, we realize that Johnston doesn't limit his discussions to the applicability of MVL to the problem of missing information solely. However, it's our position that even if MVL can be shown to be useful for other problems, it is *not* useful—in fact, it's worse than useless—for the missing information problem in particular. And it's specifically this latter notion (that MVL might be useful for dealing with the missing information problem) that has been the subject of our published criticisms; hence the emphasis on this particular issue in this response.

8. An example of an incorrect "expert" query can be found in the correspondence following publication of my column "Why Accept Wrong Answers?" in *Database Programming & Design*, 5(12): 21-22, December 1992. Alternatively, see C. J. Date, *Relational Database Writings 1991-1994*, pages 26-29. Addison-Wesley, 1995.

9. In fact, David McGoveran explicitly rebuts this claim in Part 2 of his four-part article, "Classical Logic: Nothing Compares 2 U."

10. Codd, E. F., and C. J. Date: "Much Ado about Nothing." *Database Programming & Design*, 6(10): 46-53, October 1993. Republished in C. J. Date, *Relational Database Writings 1991-1994*. Addison-Wesley, 1995.

11. Date, C. J. "EXISTS Is Not "Exists"! (Some Logical Flaws in SQL)." In C. J. Date, *Relational Database Writings 1985-1989*. Addison-Wesley, 1990.

12. Date, C. J. "The Principle of Cautious Design." In C. J. Date and Hugh Darwen, *Relational Database Writings 1989-1991*. Addison-Wesley, 1992.

13. Date, C. J. "The Default Values Approach to Missing Information." In C. J. Date and Hugh Darwen, *Relational Database Writings 1989-1991*. Addison-Wesley, 1992.

14. Codd, E. F. "Extending the Database Relational Model to Capture More Meaning." *ACM TODS*, 4(4), December 1979.

15. Codd, E. F. *The Relational Model for Database Management Version 2*. Addison-Wesley, 1990.

16. International Organization for Standardization (ISO working draft), Database Language SQL3. Document ISO/IEC JTC1/SC21/WG3 DBL OTT-003, May 1992. Also available as American National Standards Institute (ANSI) Document, ANSI X3H2 1992-109, May 1992.

17. Stoll, R. *Sets, Logic, and Axiomatic Theories*. W. H. Freeman and Co., 1961.

18. Date, C. J. "Putting the Pieces Together." *Database Programming & Design*, 7(2): 19-20, February 1994. Republished in *Relational Database Writings 1991-1994*, pages 118-119. Addison-Wesley, 1995.

C. J. DATE is an independent author, lecturer, researcher, and consultant specializing in relational database systems. Correspondence may be sent to him in care of *Database Programming & Design*, 411 Borel Ave., Suite 100, San Mateo, CA 94402.

HUGH DARWEN is a database specialist who has been with IBM United Kingdom Ltd. since 1967. He was a leading figure in the production of Business System 12 (1978-1983), an industrial-strength non-SQL relational system made for IBM's Bureau Service. In recent years, he has been an active member of the committee that drafts international SQL standards.

DAVID MCGOVERAN is president of Alternative Technologies (Boulder Creek, California), a relational database consulting firm founded in 1976. He has written numerous technical articles and is also the publisher of the "Database Product Evaluation Report Series."