

Nulls: Caveat Emptor (Let the Buyer Beware)

Tom Johnston
February, 2008.

Long, Long Ago

Some of you may remember a series of articles of mine in the old *Database Programming and Design* magazine, which ran in 1995, with one follow-up in 1998. In those articles, I debated the relative merits and deficiencies of nulls and default values with C. J. Date, David McGoveran and Hugh Darwen.

I will be asking the editor of *Intelligent Enterprise*, Mr. David Stodder, for permission to reprint these articles on this website. He was the editor of DBP&D when those articles ran, and his company is the one that still owns the copyrights.

Since that time, Date and Company have continued to argue that nulls should not be a part of the SQL language, and that default values should be used instead. In their publications, they do not reference my articles, which consequently have gradually made their way back to near total obscurity. But with Mr. Stodder's permission, I will resurrect them here because I still maintain that the position I took back then is correct, and the position(s) taken then and subsequently by Date and Company are incorrect.

The last response which Date made in that exchange is, I believe, still available in the Intelligent Enterprise archives. My final response to him is also available in those archives, and is the only one of my articles in that series that is still available on-line. Those articles, and that URL, are listed in Figure 1.

A series on two-valued logic vs. multi-valued logic, and nulls vs. default values. With several responding articles from Chris Date, David McGoveran and Hugh Darwen, which are reprinted in Date, *Relational Database Writings, 1994-1997* (Addison Wesley Longman, 1998).

“MVL: Case Open.” *Database Programming and Design*, February 1995 (vol. 8, #1), pp. 52-63.

“The Case for MVL.” *Database Programming and Design*, March 1995 (vol. 8, #3), pp. 54-68.

“More to the Point”. *Database Programming and Design*, November 1995 (vol. 8, #11), pp. 33-39.

"The Faults with Defaults." Database Programming and Design, February 1998. <http://www.dbpd.com/vault/9802xtra.htm>

Figure 1. My Contributions to a Debate about Nulls vs. Default Values.

Since Date and Company have consistently argued against the use of nulls, and in favor of the use of default values instead, one might think that my position was the obverse, an argument in favor of using nulls and against using default values. But in fact, that was not my argument then, nor is it my position today.

What Was (And Is) My Actual Position on Defaults vis-à-vis Nulls?

My position, then and now, to quote from the conclusion of the last-listed of my articles in Figure 1, was as follows:

So, with Date's new special values scheme, we have an alternative to MVL which (a) can't get equality right, (b) can't handle greater-than or less-than at all, (c) consequently returns both incorrect results and less-than-fully-informative results to queries, and (d) turns simple queries into rocket scientist queries.

Moreover, now that we understand how far from being implementable Date's scheme is, we can see that this extended debate between Date and company, and myself, has little immediate applicability to the practical work of developing databases and interfaces to them. As designers and users, we will continue to work within the constraints provided by our dialects of SQL.

The more conservative among us will avoid nulls and MVL because of understandable caution at the complexity. As I argued throughout my articles on this topic, the use of the additional inferencing power of MVL is a foray into a more complex dialogue with the database--a foray which should not be undertaken unless one is comfortable handling the complexity.

To assist the conservatives, therefore, I have sketched a two-valued logic approach to missing information that is practical, i.e. that can be used right now, with today's products. However, I claim no originality here. Indeed, I offer this sketch as little more than an existing best practice, dressed up in some fancy logical terminology, and accompanied by some explicit caveats.

As for the more adventurous among us, they will make use of nulls, as provided by our SQL dialects, and will probably get burned sometimes. But in the process, they will forge an understanding of nulls and MVL which they might not obtain from any textbook study. And so they will begin the process of rendering the complexity concomitant with the greater expressive power of nulls and MVL less daunting to us all.

Lest this "both/and" rather than "either/or" position seem rather drab and uninteresting, let me add that in my articles, I have point-by-point refutations of objections Date and Company have made to the use of nulls in the relational model – a position, by the way, which Dr. Codd himself advocated, and which one of Date's company dubbed "a *scandal of relational theory*"! (My italics. The scandal here, is on the other foot, that of someone who could make a statement like that about Dr. Codd.)

The article of mine to which there is a URL is a good example of a point-by-point refutation. That latest (to that point) position of Date's on default values took a typical approach of his, that of recommending a solution that could not be implemented in the DBMSs of that day. In fact, his recommended "solution" cannot be implemented in the DBMSs of *today*, nor does it appear on any vendor horizon that I know of, nor in a recommendation to the SQL standards committees either.

For someone whose mantra is that "theory is practice", this seems a peculiarly cavalier approach to take, one that leaves we IT professionals somewhat in the lurch.

Nevertheless, as I argue in that article, it's a good thing that Date's recommendation was never taken seriously, because it's simply wrong. I see occasional references to subsequent work on defaults vs. nulls by Date and by others of his company, including Darwen and Fabian Pascal. One day I may get around to reading their more recent material, but I can't honestly say that what I've read of their work so far, on this topic, impresses me as reflecting any depth of understanding of formal logic. Having taught propositional and predicate logic to undergraduates, I can offer the personal opinion that their work reflects a "clever undergraduate" facility with the basics, but lacks the depth of understanding that a mathematician like Dr. Codd brought to the topic.

(As for my own credentials, I do not claim to be a logician. Once upon a time – back when I was preparing for doctoral level comprehensive examinations in logic – I was able to struggle through most articles in such journals as the *Notre Dame Journal of Formal Logic*. But that by no means made me a logician. It just made me informed enough to realize the gap between my knowledge and that of a logician, a gap about the same size, it seems to me, as the gap between a clever undergraduate's understanding and my own.)

Until I have been able to reprint my articles here, I'd like to return to the discussion by warning my readers of the dangers involved in using nulls, something Date and Company have been doing for years. For, with them, I believe, and have always said, that today's implementations of nulls in SQL – and I do mean "implementations" in the plural, since

Nulls. Caveat Emptor.

© Copyright 2008, Dr. Tom Johnston.

Only personal, non-commercial copies, are permitted.

Page 1.

implementations do in fact vary from one vendor's SQL to another's – are all flawed, confusing, confused and potentially misleading!

Nulls: Proceed With Caution.

But before I proceed to my own material, let me reference one of the best detailed explanations of the vagaries of SQL nulls that I have come across. It's by a guy identified only as "Shahid", and appeared in April 2006 on the website www.healthcareguy.com. The full URL is: www.healthcareguy.com/index.php/archives/236.

As I said, I think this is an excellent article, better than anything I could have written myself. If, after reading and understanding this article, you still want to make limited use of SQL nulls in your databases, then I think you will be well prepared to do so.

Now for my own material. It's origins are in a position paper taken by a client on mapping from source systems into a data warehouse, and in particular how to handle blanks, zeroes and nulls in both the source data and in its representation in the target data warehouse. It discusses just one of the ways in which using nulls can mislead those who view and interpret result sets over data which includes one or more nulls. I present this material in the form of a dialog between myself (Tom) and a colleague (Colleen).

A Dialog About SQL Nulls.

Tom.

So you want to map zeroes in source data into nulls, as you load that data into the data warehouse. Well, I just found a nice article in Wikipedia that points out some of the problems in replacing zeroes with nulls.

To paraphrase that article, consider the following table. (The following paragraph may actually be a direct quote from that article.)

I	J
150	150
200	200
350	350
NULL	0

The SQL AVG aggregate function returns 233 when applied to column I, but returns 175 when applied to column J. The aggregate function's Null-elimination step accounts for the difference in these results. The only aggregate function that does not implicitly eliminate Null is the COUNT(*) function.

Nulls. Caveat Emptor.

© Copyright 2008, Dr. Tom Johnston.

Only personal, non-commercial copies, are permitted.

Page 1.

Business users, interpreting statistical function results, often do not realize this problem. Note that the only difference between the two columns is NULL vs. zeroes. The result can easily be a wrong decision. I'll elaborate a little.

So suppose the sample data from this Wikipedia article on NULLs and SQL was being input into our data warehouse. Now suppose that input data contained a NULL, as shown in column I, but that we have decided to convert all such NULLs to zeroes. This effectively means that column I was input, and column J was stored in the warehouse. Doing an AVG on the source data, we would get 233. Doing an average on the corresponding warehouse data, we would get 175.

Nothing in the real world is different corresponding to this difference in results. The only difference, in results that can be as different as you care to imagine, and that can be as critical to business decisions as you care to imagine, is our policy about what to do with NULLs in data being loaded into the warehouse.

Well, suppose everyone knows that this is what we are doing, converting nulls in numeric data to zeroes as we load that data into the warehouse. Does this mean that we don't have a problem anymore?

Not at all. In fact, we now have two problems.

- First, some source data, we may suppose, contained zeroes as a real value. We are now unable to distinguish real value zeroes from null-replacement zeroes. Since nulls, representing a lack of information, introduce uncertainty into any statistical results they contribute to, this inability to distinguish real from null-substitute zeroes means that we don't know how much uncertainty there is in our statistical results.
- The second problem is that 175 is just as likely to be an incorrect answer as in 233. All we've done is muddy things up.

A middle-way alternative is for each numeric source column that may contain NULLs, matched to a warehouse column that will contain zeroes instead, we add an extra one-byte column with a code explaining that the column it describes (a) was zeroes in the original input data, or (b) was null and was then converted to zeroes. Something like this is probably the best solution to proceed with, but notice that it is a lot of hard work.

Colleen.

If someone asks me what my average speed is, why would I want to add a zero if the real value for the fourth sample should be 'no value submitted'? It seems to me that 233 is the correct answer. Now if one time period actually reported a speed of 0, then the average should be 175.

So, why would I want zeros instead of nulls?

Nulls. Caveat Emptor.

© Copyright 2008, Dr. Tom Johnston.

Only personal, non-commercial copies, are permitted.

Page 1.

If the problem is that business users want it dumbed down - then they need to say so, and they need to stand up and accept the consequences of their choices. If the problem is that our developers don't know how to code with nulls, then they need training or we need new developers with skills that meet our requirements.

Tom.

OK, let's assume that these two columns are reporting time trials, and each entry is an average speed for one run.

Regardless of interpretation, the correct answer is *not* 233. The correct answer is: we don't know what the average is, because we are missing some information needed to compute it.

The only case where 233 would be the correct answer is the one case where the fourth speed was 232. In that case, we would have $(150 + 200 + 350 + 232) / 4$, which is 233. If we are talking about cars with a top speed of 350mph (and all speeds reported in whole numbers), let's say, then the odds are 350 to 1 that 232 is the wrong fourth speed, and also that 233 is the wrong average over all four time trials.

Colleen.

I would say that the business question would most likely be - "of the ones reporting, what is the average?" The business might want to know the ratio of reported to not reported, but statistically, unreported values most likely will not skew the average of the reported rows, if the ratio of reported rows is high enough. I would say that this is more a user education issue - they need to ask the right question - before anyone goes over to the computer and starts calculating. Tech people want to show they can interact with the computer - the problem is, often they can not interact with the person who needs information from the computer.

Of course the 'theoretical' answer is that until everyone reports a value, we can't calculate an average. However, you and I both know that the business knows that is unrealistic.

Tom.

OK, let's say that what the business really wants to know is "Of the ones reporting, *and* providing all required data, what is the average." Maybe that's what they want. But I don't think so. Because the more events report in with NULL (and are ignored, on your assumption), the less reliable those results are. And the business should be informed of what's going on.

For example, if there were 100 time trials reported, but an actual speed provided for only three of them, what is the range of values that the average could take on, if the missing

values were provided? It's obviously from a low of $(150 + 200 + 350 + 0 + 0 + \dots + 0)/100$, to a high of $(150 + 200 + 350 + 350 + 350 + \dots + 350)/100$. That works out to a range from 7mph to 247mph (rounded up). So the odds of any SQL computed average being correct is 240 to 1 against!

Clearly, the business should be informed of the difference between a rock-solid 233 and a wild-ass guess 233. Since an average is a sum divided by a count, and NULLs are ignored in sums but not in counts, the result is uncertain in direct proportion to the number of NULLs involved. If we replace NULLs by zeroes, we get a different answer. But that answer is no more reliable or valid than the other one.

Tom (continuing on).

Letting the user know how reliable a computed number is, is not a matter of educating the user. It's a matter of telling them how reliable the number is.

Assuming that there won't be enough missing information to be concerned about isn't a very good idea either.

I can't argue with further waffling about what's reasonable to assume, or what we believe users actually do assume. On either approach, a definite answer is given to the user. On both approaches, the answer is very likely wrong. And nobody's talking about saying anything about pointing that out to the users.

Colleen.

If I reported a portion of the data, but not all of the data - did I report? - yes, then - was it a complete report? - well that depends on your definition of complete. Maybe that field is not 'required'; optional fields are just that, optional. If it is required and I don't report it, then the 'default' might be the average of all which were recorded up to that point. Or the 'default' might be to record the partial row, but put in 'pending' status because it is incomplete.

Really, I understand how the recorded data looks different and calculates differently depending on the use of defaults and nulls. I really have written enough SQL to understand that portion of the problem. I am trying to shed light on the implications, from a business perspective and not a technical one. I believe the real issue is 'truth in recording'; data defaults are from the flat file era when nulls were not even invented. Do we not have a burden to help bring people into a more technically mature world? If they were doing this paper based, what would they do if someone did not report a fact? That is the real question. I suspect the answer depends on 'what column is it?' - and how many different flavors of replies are there based on the different columns. So, swooping default values statements seem insufficient to me.

Nulls. Caveat Emptor.

© Copyright 2008, Dr. Tom Johnston.

Only personal, non-commercial copies, are permitted.

Page 1.

Tom - your example is correct - but remember - asking a single question of data is a "buyer beware" type environment. If someone is foolish enough to ask what is the average - and there are 100 to 10,000 rows, and the answer is 233, and they don't ask how many rows participated in the calculation - then I still maintain that the user is at fault, not the computer. It is a user education issue, computers are really, really dumb - and they will only answer the question asked - lawyers would love if people acted the same way while on the witness stand. The computer's position is 'asked and answered'. Expecting the computer to compensate for user ignorance seems like asking for trouble. Remember, not all people are ignorant in like manner.

Tom.

I'm not forgetting anything. I'm just pointing out the problems with the current SQL implementation of NULLs. Chris Date would be proud of me!

Current best practice is not good practice. It's bad practice which happens to be current best practice. It will remain best practice until I or someone else persuades a major enterprise to manage unknown data in a clear and consistent way.