To: ttm@thethirdmanifesto.com
From: Derek Asirvadem <derek.asirvadem@gmail.com>
Subject: Fwd: Re: Re: [TTM] A joyful experience with Rel (and DKNF)
Cc:
Bcc:

> From: Hugh Darwen <hughdarwen@gmail.com>

I intend no disrespect to you Hugh, and this was not directed at or to you. Rather at the comments made.

>> On 30 January 2012 14:58, Derek Asirvadem <derek.asirvadem@gmail.com> wrote:
>>
>> [...] There is no dispute that the SQL definition famously has The Null Problem. Half that problem is in the eye of the beholder, particularly if they stick to the definitions and cannot write program code. For the vendors who have implemented SQL as a programing language (well, programmers need predictable execution), they have had to implement consistency which does not exist in the SQL definition. So there are clearly three categories:
>>
>> 1 SQL Definition, written by a bunch of non-technicians who expended more energy fighting each other than in serving the community, and the famous Null Problem. Good for arguments about how stupid SQL (in definition, not in implementation). Nothing to do with reality.
>
> SQL's 3VL appeared in the original products, having been proposed by E.F. Codd. The SQL standard has been drafted exclusively by technicians (such as myself), nearly all representing vendors such as IBM and Oracle. Those technicians try to serve both the user community and the vendor community by offering a common definition for all vendors to adhere to. The fights arise when one vendor has implemented, or plans to implement, a certain feature in a different way from the other.

I have not seen any evidence of that.

The mess that SQL is; the various contradictions; the onerous syntax; the total failure of progressive versions to overcome the blunders of previous versions; and allowances for doing things in two contradictory ways, the 3VL, all stand as de facto evidence of the statement I made.

>> 2 The big iron class of SQL implementors, who implemented a consistent null handling. Yes, Null = Null and does not equal anything else; yes, Null is the Unknown Value, because we need it placed in the spectrum of Known Values, in order to write simple program code. DB2 and Sybase have an "ANSI Null Handling" session-level switch, and the default is of course OFF.If one really wants the insanity of [1], one merely sets the switch ON.
>
> Wow! Is that right?

That, among the various other statements you have made about what SQL is and is not, contradicts the veracity of your statements above. Or else, accepting that you know SQL in the definition committee sense, we have to note that you do not know SQL in the implementation sense (both the requirements of the implementing programmer and the vendor implementing the platform) where both of these are (a) correct and (b) successful, in the sense that there are no errors for years afterwards, zero maintenance code and databases (as opposed to market success of abominations such as Oracle and PostGre-non-SQL).

(I do accept that you were on the SQL [definition] committee.)

I would not wish to make determinations re someone such as you, but in my book, and specifically in the context in which I use the word **technician** above, observing all of your writings here, that I have read, you come across strongly and consistently as an academic (non-technician). Your definition of the term is clearly quite different from mine, and what I meant when I used it.

(And please , let's not be drawn into inferences from that statement, other than the context in the original post and here.)

> "Session switches" are truly dreadful for such purposes. Here I could have an update that either fails or succeeds according to the setting of the switch! That's because a table constraint such as X > Y succeeds if it evaluates to UNKNOWN.

Here is an example of what I mean, evidence for my statements above. That is a typical non-technical (academic) response: it is correct, but irrelevant in the real physical technical world.

The simple fact of any session level setting (and in general, any configuration option), is that:

• the setting has a default value, OFF

• one only needs to change it if one wants the non-default behaviour

----

• there is no "purpose"

(or your purpose is to remain in confusion and unpredictability, and ours is to eliminate it; having eliminated it, we need not keep mentioning it)

(Obviously no offence intended. The analogy that comes to mind is, we have nailed the cellar window shut, to keep the mice out; we do not need to get up in the middle of the night and check the nailed window, nor the larder. We sleep very soundly, and we have uncontaminated bread and cheese for breakfast.
We do however accept that there are people who do not believe in nails, who live in a frightening and hungry world, who keep getting up in the middle of the night to check the window, and the larder.
They are occupied in fulfilling their basic survival needs.
We have no need to convert their beliefs.
We extend the courtesy of listening patiently to *their* warnings about *their* food and *their* plundering mice, which do not exist in *our* world.
We have no problem with such people.
The only problem occurs when they try to impose *their* neuroses and sufferings on us, and tell how restless *our* sleep is; how frightening *our* world is; how empty *our* breakfast plates are.
We have nails.
After breaking our fast, we are occupied in building better, stronger, cheaper, nails.)

• no one in their right mind would `SET ANSI NULL ON`

• (and if they did, it would fail the standards implemented in most organisations, and thus be disallowed in production)

• 99% of the people in the Sybase community do not know the session level setting exists; they only know the consistent and predictable behaviour of Sybase (and DB2) Null. (No comment on the sufferings of Oracle and PostGre-non-SQL programmers.)

• I have never seen a technician use the ON setting

• the issue (your suggested "problem" with SQL Null) has never come in hundreds of technical conversations and discussions and consultations and education sessions that I have engaged in (counting of course only that which included the subject of Null)

• the only time I have actually seen it, was in code written by an academic, which was one of those "poof of concept" projects that did not attract approval or funding. The academic was a "famous Ingres person" whom I had not heard of. The customer valued his "concept" but not his code or database or demands (he had various obscure demands, in addition to ASNI NULL), so I replaced his 400+ lines of non-compliant, unacceptable code with 30 lines of compliant code, to keep the concept around, but in a useable condition. His "database" was eliminated because I could read everything he needed directly from production

> and did not try to replicate the intermediate "database" that he required (I did not understand his "need" for his "database", and after reading his "database", believe me, I did not *try* to understand it).

Therefore the issue is entirely academic, and that is again proved by this exchange, which is academic. It is a *total and complete* non-issue in the real world.

The point being, absolutely everyone I have ever worked with, every code segment I have ever dealt with, in the Sybase world (and lesser volume in the DB2 world), has never used ANSI NULL and all its commensurate insanity. They do not suffer the academic problem you describe (quite correctly in the academic sense; and completely irrelevant in the technical sense).

They would be bored to stone in 30 seconds flat if anyone tried to inform them of the dangers of a setting they (99%) do not know about or care about. The one percent would reach snoring stage at about 3 minutes, and stone at 5 minutes. They are technicians interested in facts regarding the physical world, they have no interest in hypothetical "problems" that have no possibility of occurring in their world. I am probably the only one who takes the time, to extend the courtesy, to answer your point.

Such is the value of a good vendor, who has:

a. worked through the insanity produced by the evidently non-technical SQL committee, and provided a 100% consistent, useable, and predictable implementation of Null in SQL

b. within the spirit and meaning of Codd's work

c. and additionally provided a 100% ANSI compliant behaviour for those programmers who are attached to their unpredictability (and consequently more complex code), or those box-checkers (accountants parading themselves as auditors, whose tick mark we need, but for whom we have no time to educate) who need to check the "ANSI SQL Null Behaviour Compliant" box.

Eg. such vendors do not take the time to answer your point, re your "problem" with SQL, they have written the "problem" out of the product. As I and many (not all) architects and modellers do, in the implementation of a database or code.

> Here I could have an update that either fails or succeeds according to the setting of the switch! That's because a table constraint such as X > Y succeeds if it evaluates to UNKNOWN.

(Assuming you mean under theoretical SQL and not a real SQL) Really ? Ok, fine, that may be a possibility in the hypothetical universe. But I can't be asked to contemplate it.

Hang on. Even if that could be true, it would only be true, if the programmer were moronic enough to keep switching the setting back and forth. So I would say to that programmer (or "user" or munchkin or whatever), what happens if you envelope the constraint in a RAND() function, does it get more or less interesting ? And then, in consideration of my responsibilities to the organisation that is paying my fees, take steps to ensure he never works for the IT department. If his qualifications allude to "engineer" or "technician", I would have them scrutinised by HR.

> (Btw, what do 1 > NULL and NULL > 1 evaluate to?)

Before I try it, please do recall my post.

>> 2 The big iron class of SQL implementors, who implemented a consistent null handling. Yes, Null = Null and does not equal anything else; yes, Null is the Unknown Value, because we need it placed in the spectrum of Known Values, in order to write simple program code. DB2 and Sybase have an "ANSI Null Handling" session-level switch, and the default is of course OFF. If one really wants the insanity of [1], one merely sets the switch ON.

Techincians and implementors (the ones that are worth considering, who implement standards and write unbreakable code, and ones like me, who write unbreakable databases) need consistency and predictability in an SQL product. No time-wasting puzzles or figuring out behaviour under different circumstances. We code in a real product, we do not contemplate the hypotheical that is outside our real world.

As technicians who do *not* have to read manuals more than once, we *know* that *by definition*, Null is **The Unknown Value**. Therefore without having to open an SQL window and type commands into it, and actually *try* it, we *know* that any known or finite value compared to The Unknown value would yield "false". And we have been coding on that basis (one that we can rely upon) for decades. And that such code behaves as predicted; never fails in UAT; never changes behaviour; etc. And we have the consequent confidence of that, that come from decades of consistent and uneventful behaviour.

I suppose you would like me to try it anyway. Let me get a coffee in order to overcome this sudden feeling of stone ...

Ok, I am back. This:

```
    IF NULL > 1 PRINT "Null is greater than 1"
    ELSE PRINT "Null is not greater than 1"
    IF 1 > NULL PRINT "1 is gweater than Null"
    ELSE PRINT "1 is not greater than Null"
```
yields:
```
    Null is not greater than 1
    1 is not greater than Null
```

Forgive me, I have already extended courtesy to you, I have run out of coffee, and I can't be asked to `SET ANSI NULL ON.`

I do not like that as the last word, instead I would like the context of my original post restated:

> There is no dispute that the SQL definition famously has The Null Problem. Half that problem is in the eye of the beholder, particularly if they stick to the definitions and cannot write program code. For the vendors who have implemented SQL as a programing language (well, programmers need predictable execution), they have had to implement consistency which does not exist in the SQL definition. So there are clearly three categories:
>
> 1 SQL Definition, written by a bunch of non-technicians who expended more energy fighting each other than in serving the community, and the famous Null Problem. Good for arguments about how stupid SQL (in definition, not in implementation). Nothing to do with reality.
>
> 2 The big iron class of SQL implementors, who implemented a consistent null handling. Yes, Null = Null and does not equal anything else; yes, Null is the Unknown Value, because we need it placed in the spectrum of Known Values, in order to write simple program code. DB2 and Sybase have an "ANSI Null Handling" session-level switch, and the default is of course OFF.If one really wants the insanity of [1], one merely sets the switch ON.
>
> 3 The idiot class of SQL implementors, Oracle, the freeware non-SQLs passing themselves off as SQL, etc. Various levels of inconsistent Null (a) determination and (b) Null handling.
>
> And all that is separate to what I call the Introduced Null Problem, to differentiate it from the better-known and more famous Null Problem (three implementation categories above), which is the implementation of Nulls in data collections, due to lack of Normalisation or modelling errors, where Normalisation and absence of modelling errors would result in dramatically less Nulls in the database.
>
> For those of us who live in [2], we have no restrict problems with Null; we have very small code segments to handle Null (and they operate consistently). And if we can Normalise and Model, yes, well we have no Nulls at all in the db.

> That is not to say that we [2 again] never write code using "NULL". We do. All the time. because we are coding in SQL, and can never really get away from The Null. Sometimes we use it because it is convenient. But always we use it because it is predictable, and easy. Which is a far cry from the picture painted on this list. Which is really about [1] and the worse [3]s, not the happy campers in [2].

--
Regards
Derek Asirvadem
Faithful RM Stalwart, Standard-compliant Practitioner
http://www.softwaregems.com.au