Are nulls in a relational database okay? [closed]

Asked 16 years, 2 months ago Modified 1 year, 1 month ago

Viewed 43k times

82

Closed. This question needs details or clarity. It is not currently accepting answers.

Want to improve this question? Add details and clarify the problem by editing this post.

Closed 4 years ago.

Improve this question

There's a school of thought that null values should not be allowed in a relational database. That is, a table's attribute (column) should not allow null values. Coming from a software development background, I really don't understand this. It seems that if null is valid within the context of the attribute, then it should be allowed. This is very common in Java where object references are often null. Not having an extensive database experience, I wonder if I'm missing something here.

Are nulls in a relational database okay?

database-design oop null

Share

Improve this question

Follow

edited Nov 2, 2023 at 8:33

community wiki
4 revs, 4 users 67%
philipxy

If we shouldn't use NULL, why would RDBMSs allow us to use NULL at all? There's nothing wrong with NULL as long as you know how to deal with them. Creating separate tables to store columns with null values in every scenario is overly misguiding. – Fr0zenFyr Aug 13, 2013 at 7:31

33 Answers

Sorted by: Highest score (default)

1 2 Next

80

Nulls are negatively viewed from the perspective of database normalization. The idea being that if a value can be nothing, then you really should split that out into another sparse table such that you don't require rows for items which have no value.

It's an effort to make sure all data is valid and valued.

In some cases having a null field is useful, though, especially when you want to avoid yet another join for

performance reasons (although this shouldn't be an issue if the database engine is setup properly, except in extraordinary high performance scenarios.)

Share Improve this answer Follow

edited Nov 2, 2023 at 8:36

community wiki
2 revs, 2 users 86%
Adam Davis

i cannot fathom where this perception 'nulls are negatively viewed...' originated, can you please provide a reference? – Steven A. Lowe Oct 6, 2008 at 15:04

1 You cannot be in first normal form with nullable columns. One reference that states this explicitly is en.wikipedia.org/wiki/Database_normalization#First_normal_form "Simply put, a table with a unique key and without any nullable columns is in 1NF." – Adam Davis Oct 6, 2008 at 17:03

1 Okay, but then I would counter that ensuring that your DB is in 1NF isn't necessarily all that valuable, especially if it's going to cost you usability and clarity. (Don't get me wrong, normalization is, of course, generally desirable...but adhering to it to the point of eliminating nulls seems counterproductive.) – Beska Apr 22, 2009 at 15:25

4 References: CJ Date is pretty well known in relational databases. He's the chief proponent of "nulls considered harmful" e.g. see here dcs.warwick.ac.uk/~hugh/TTM/Missing-info-without-nulls.pdf – MarkJ Dec 11, 2009 at 1:13

8 I'm genuinely curious, when is a database over normalized? Relational design cannot contain null values, if a sparse table

pattern eliminates null's and keeps the database purely relational, what is the issue? People mention joins but I'd challenge that notion since a relation with two tuples takes almost nothing to join to a base table, even at extremely high loads. Surely the impact is only on design productivity? People don't normalize databases because at some point it becomes much harder to design and query. Although the effort should be improving this, not breaking relational principles. – npeterson Apr 9, 2012 at 19:03

46

One argument against nulls is that they don't have a well-defined interpretation. If a field is null, that could be interpreted as any of the following:

- The value is "Nothing" or "Empty set"
- There is no value that makes sense for that field.
- The value is unknown.
- The value hasn't been entered yet.
- The value is an empty string (for databases that don't distinguish between nulls and empty strings).
- Some application-specific meaning (e.g., "If the value is null, then use a default value.")
- An error has occurred, causing the field to have a null value when it really shouldn't.

Some schema designers demand that all values and data types should have well-defined interpretations, therefore nulls are bad.

Share Improve this answer Follow

edited Sep 20, 2013 at 14:19

community wiki
4 revs
Kristopher Johnson

6 An integer doesn't have a well defined meaning either. But there is nothing to prevent you from adding one via documentation. – Jonathan Allen Oct 6, 2008 at 5:18

1 Another meaning is "Oops, my process failed to populate the field w/ the intended value". For fields that are FKs to a set of enumerated values, one can add a representation of NULL to that primary table. With this technique, you can still allow the "no data" concept, yet be explicit about it – 6eorge Jetson Oct 6, 2008 at 6:09

1 +1 because this is the famous argument against nulls in database schemas as promulgated by CJ Date (I don't necessarily agree with it) e.g. his book *Introduction to database systems* – MarkJ Dec 11, 2009 at 1:17

NULL means "we don't have this value". In most cases we don't need to know anything further about why the value is not there, just as we don't need to know who entered a particular value and when, nor whether a value is expected to change in future, nor whether a value is certain or uncertain. As a developer, I would much rather deal with nullable fields (when necessary) than the complexity of a proliferation of unnecessary tables. – Sam Watkins Jan 17, 2017 at 7:31

1 His solution(s) doesn't solve the problem either. If a "side"-table record is nonexisting, is the value missing, unknown, error occured or something else. Plus, you can always waste another field storing "type of missing record", if this information is interesting to someone. This begs the question though, if that field is also NULL, does one need another field

to explain the absense of *that* data too – siggemannen Aug 17, 2023 at 16:34

29

It depends.

As long as you understand why you are allowing `NULL`s in the database (**the choice needs to be made on a per-column basis**) AND how you will interpret, ignore or otherwise deal with them, they are fine.

For instance, a column like `NUM_CHILDREN` - what do you do if you don't know the answer - it should be `NULL`. In my mind, there is no other best option for this column's design (even if you have a flag to determine whether the `NUM_CHILDREN` column is valid, you still have to have a value in this column).

On the other hand, if you don't allow `NULL`s and have special reserved values for certain cases (instead of flags), like -1 for number of children when it is really unknown, you have to address these in a similar way, in terms of conventions, documentation, etc.

So, ultimately, the issues have to be addressed with conventions, documentation and consistency.

The alternative, as apparently espoused by Adam Davis in the above answer, of normalizing the columns out to sparse (or not so sparse, in the case of the `NUM_CHILDREN` example or any example where most of the data has

known values) tables, while able to eliminate all NULLs, is non-workable in general practice.

In many cases where an attribute is unknown, it makes little sense to join to another table for each and every column which could allow `NULL`s in a simpler design. The overhead of joins, the space requirements for theprimary keys make little sense in the real world.

This brings to mind the way duplicate rows can be eliminated by adding a cardinality column, while it theoretically solves the problem of not having a unique key, in practice that is sometimes impossible - for instance, in large scale data. The purists are then quick to suggest a surrogate PK instead, yet the idea that a meaningless surrogate can form part of a tuple (row) in a relation (table) is laughable from the point of view of the relational theory.

Share Improve this answer Follow

edited Oct 2, 2008 at 17:29

community wiki
2 revs
Cade Roux

23

There are several different objections to the use of NULL. Some of the objections are based on database theory. In theory, there is no difference between theory and practice. In practice, there is.

It is true that a fully normalized database can get along without NULLS at all. Any place where a data value has to be left out is a place where an entire row can be left out with no loss of information.

In practice, decomposing tables to this extent serves no great useful purpose, and the programming needed to perform simple CRUD operations on the database become more tedious and error prone, rather than less.

There are places where the use of NULLS can cause problems: essentially these revolve around the following question: what does missing data really mean? All a NULL really conveys is that there is no value stored in a given field. But the inferences application programmers draw from missing data are sometimes incorrect, and that causes a lot of problems.

Data can be missing from a location for a variety of reasons. Here are a few:

1. The data is inapplicable in this context. e.g. spouse's first name for a single person.
2. The user of a data entry form left a field blank, and the application does not require an entry in the field.
3. The data is copied to the database from some other database or file, and there was missing data in the source.
4. There is an optional relationship encoded in a foreign key.

5. An empty string was stored in an Oracle database.

Here are some guidelines about when to avoid NULLS:

If in the course of normal expected programming, query writers have to write a lot of ISNULL, NV, COALESCE, or similar code in order to substitute a valid value for the NULL. Sometimes, it's better to make the substitution at store time, provided what's being stored is "reality".

If counts are likely to be off because rows containing a NULL were counted. Often, this can be obviated by just selecting count(MyField) instead of count(*).

Here is one place where you by golly better get used to NULLS, and program accordingly: whenever you start using outer joins, like LEFT JOIN and RIGHT JOIN. The whole point behind an outer join as distinct from an inner join is to get rows when some matching data is missing. The missing data will be given as NULLS.

My bottom line: don't dismiss theory without understanding it. But learn when to depart from theory as well as how to follow it.

edited Apr 22, 2009 at 20:05

community wiki
2 revs
Walter Mitty

2 Your case goes beyond the original question. You might want to research "fourth normal form" – Walter Mitty Dec 7, 2013 at 16:50

Null markers are fine. Really, they are.

22

Share Improve this answer Follow

edited Oct 2, 2008 at 17:06

community wiki
2 revs, 2 users 67%
Ken Wootton

1 like most features of anything, nulls are fine only if you know how to use them. Remember that it does require another bit of storage for every row * every column that enable NULL. – Dvir Berebi Dec 25, 2010 at 20:27

53 Without an explanation, this answer may become useless in case if someone else posts an opposite opinion. For example, if someone posts a claim like *"Null markers aren't fine. Really, they aren't."*, how would this answer help reader to pick of two opposing opinions? Consider editing it, to better fit How to Answer guidelines – gnat Jun 22, 2015 at 17:04

This does not explain what a "marker" is. (And it is much simpler to clearly & correctly address null semantics by simply using the fact that null is a value that is treated specially by SQL syntax & operators--pace SQL & null apologist rhetoric.) – philipxy Mar 10, 2019 at 22:44

18

There is nothing wrong with using NULL for data fields. You have to be careful when setting keys to null. Primary keys should never be NULL. Foreign keys can be null but you have to be careful not to create orphan records.

If something is "non existent" then you should use NULL instead of an empty string or other kind of flag.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:02

community wiki
Ken

2 "You have to be careful when setting keys to null...." A Primary Key column can *never* be NULL. Any column that is part of a primary key can never be NULL. – Taptronic Oct 2, 2008 at 17:11

More or less supporting what you said for emphasis. ;-) – Taptronic Oct 2, 2008 at 17:12

4 "If something is "non existent" then you should use NULL instead of an empty string or other kind of flag." This bears repeating – Bob Probst Oct 2, 2008 at 17:19

9 If something is missing, there should be a missing row in some table. "NULL" is not "missing", "NULL" is "anything". This bears repeating. – Constantin Oct 14, 2008 at 8:44

3 If something is missing, there should be a missing row in some table. "NULL" is not "missing", "NULL" is "anything". This bears repeating. (repeated) – simon Nov 4, 2009 at 15:31

15

Instead of writing up all the issues of NULL, and tristate vs boolean logic, etc. - I'll offer this pithy advice:

1. Don't allow NULL in your columns, until you find yourself adding a magic value to represent missing or incomplete data.
2. Since you're asking this question, you should be *very* careful in how you approach NULL. There's a lot of nonobvious pitfalls to it. When in doubt, don't use NULL.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:11

community wiki
Mark Brackett

10

There is another alternative to using "N/A" or "N/K" or the empty string - a separate table.

E.g. if we may or may not know a customer's phone number:

```
CREATE TABLE Customer (ID int PRIMARY KEY, Name
varchar(100) NOT NULL, Address varchar(200) NOT
NULL);
CREATE TABLE CustomerPhone (ID int PRIMARY KEY,
Phone varchar(20) NOT NULL, CONSTRAINT
FK_CustomerPhone_Customer FOREIGN KEY (ID)
REFERENCES Customer (ID));
```

If we don't know the phone number we just don't add a row to the second table.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:18

community wiki
finnw

8

Don't underestimate the complexity you create by making a field NULLable. For example, the following where clause looks like it will match all rows (bits can only be 1 or 0, right?)

```
where bitfield in (1,0)
```

But if the bitfield is NULLable, it will miss some. Or take the following query:

```
select * from mytable
where id not in (select id from excludetable)
```

Now if the excludetable contains a null and a 1, this translates to:

```
select * from mytable
where id <> NULL and id <> 1
```

But "id <> NULL" is false for any value of id, so this will never return any rows. This catches even experienced database developers by surpise.

Given that most people can be caught off-guard by NULL, I try to avoid it when I can.

Share Improve this answer Follow

edited Apr 22, 2009 at 16:59

community wiki
2 revs
Andomar

> Bugs and surprises are inevitable in programming. In my experience, strictly avoiding NULLs leads to much more complex database designs with many more tables. Allowing NULL when needed is comparatively less difficult and error prone. – Sam Watkins Jan 17, 2017 at 7:43

7

I would say that Nulls should definitely be used. There is no other right way to represent lack of data. For example, it would be wrong to use an empty string to represent a missing address line, or it would be wrong to use 0 to represent a missing age data item. Because both an empty string and 0 are data. Null is the best way to represent such a scenario.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:05

community wiki
Vaibhav

1 "Null is the best way to represent such a scenario." I disagree. Given (first_name, middle_initial, last_name), what does NULL in middle_initial mean? It's not clear; either we don't know or it doesn't exist. NULL doesn't tell us which.
– Dave Oct 2, 2008 at 17:25

7 And if we don't know is it because we haven't asked, or they've refused to reveal it. And If the latter, is it because of shame or spite? we can't say. If it's important to your app to know the difference then you can store the reason somewhere else. If it's not import, who the hell cares?
– Mark Brady Oct 2, 2008 at 17:35

You could have an Address table and not have anything in there for the link to the Person table. I like that better.
– Joe Phillips Oct 2, 2008 at 17:56

1 It is not true that there is "no other right way to represent lack of data". Indeed, according to relational algebra, using nulls is incorrect. The correct way is to have separate tables for each optional field, as Cade suggests. As others have pointed out, this rapidly becomes unwieldy. – Dour High Arch Oct 3, 2008 at 16:54

2 In Oracle, an empty string, is in fact, NULL :)
– Camilo Díaz Repka Oct 6, 2008 at 4:49

This is a huge can of worms, because NULL can mean so many things:

- No date of death because the person is still alive.
- No cell phone number because we don't know what it is or even if it exists.
- No social security number because that person is know to not have one.

Some of these can be avoided by normalisation, some of them can be avoided by the presence of a value in that column ("N/A"), some of them can be mitigated by having a separate column to explain the presence of the NULL ("N/K", "N/A" etc).

It's also a can of worms because the SQL syntax needed to find them is different to that of non-null values, it's difficult to join on them, and they are generally not included in index entries.

Because of the former reason you're going to find cases where a null is unavoidable.

Because of the latter reason you should still do your best to minimise the number of them.

Regardless, always use NOT NULL constraints to guard against nulls where a value is required.

edited Oct 2, 2008 at 17:15

community wiki
2 revs, 2 users 93%
David Aldridge

A good argument for allowing reserved values for columns outside the normal range of the column. Would allow us to have a variety of self-documenting flexibility in column design with constants like enums to represent "UNKNOWN", "NO DEATH DATE", etc. without endless constraints and flags.
– Cade Roux Oct 3, 2008 at 3:10

1 NULL just means one thing: "we don't have this data". If you need more detailed explanation for that (and it's usually NOT necessary), you can add more columns to explain it.
– Sam Watkins Jan 17, 2017 at 7:45

@SamWatkins I think we mean "mean" in two different ways there. – David Aldridge Jan 17, 2017 at 13:44

6

The main issue with nulls is that they have special semantics that can produce unexpected results with comparisons, aggregates and joins.

- Nothing is ever equal to null, and nothing is ever not equal to, greater than or less than null, so you have to set nulls to a placeholder value if you want do any bulk comparison.
- This is also a problem on composite keys that might be used in a join. Where the natural key includes a nullable column you might want to consider using a synthetic key.

- Nulls can drop out of counts, which may not be the semantics you desire.
- Nulls in a column that you can join against will eliminate rows from an inner join. In general this is probably desired behaviour, but it can lay elephant traps for people doing reporting.

There are quite a few other subtleties to nulls. Joe Celko's SQL for Smarties has a whole chapter on the subject and is a good book and worth reading anyway. Some examples of places where nulls are a good solution are:

- Optional relationships where a joined entity may or may not be present. Null is the only way to represent an optional relationship on a foreign key column.
- Columns that you may wish to use to null to drop out of counts.
- Optional numeric (e.g. currency) values that may or may not be present. There is no effective placeholder value for 'not recorded' in number systems (particularly where zero is a legal value), so null is really the only good way to do this.

Some examples of places where you might want to avoid using nulls because they are likely to cause subtle bugs.

- 'Not Recorded' values on code fields with a FK against a reference table. Use a placeholder value, so you (or some random business analyst down the

track) don't inadvertently drop rows out of result sets when doing a query against the database.

- Description fields where nothing has been entered - null string (`''`) works fine for this. This saves having to treat the nulls as a special case.
- Optional columns on a reporting or data warehouse system. For this situation, make a placeholder row for 'Not Recorded' in the dimension and join against that. This simplifies querying and plays nicely with ad-hoc reporting tools.

Again, Celko's book is a good treatment of the subject.

Share Improve this answer Follow

answered Apr 22, 2009 at 15:38

community wiki
ConcernedOfTunbridgeWells

5

The best thing to know about normal forms is that they are guides and guides should not be doggedly adhered to. When the world of academia clashes with the actual world you seldom find many surviving warriors of academia.

It's ok to use nulls. Just evaluate your situation and decide if you want them to show up in the table or collapse the data into another related table if you feel your ratio of null values to actual values is too high.

As a friend is fond of saying, "Don't let the perfect be the enemy of the good". Voltaire also said that.

Share Improve this answer Follow

edited Nov 2, 2023 at 8:42

community wiki
3 revs, 2 users 60%
ScottCher

4

According to strict relational algebra, nulls are not needed. However for any practical project, they are needed.

First, much real-world data is unknown or not applicable and nulls implement that behavior well. Second, they make views and outer joins much more practical.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:03

community wiki
Dour High Arch

3

You'll find with step-by-step data acquisition systems that you can't avoid having nulls in a database because the order of asking questions / data gathering very rarely matches the logical data model.

Or you can default the values (requiring code to handle these default values). You can assume all strings are empty instead of null, for example, in your model.

Or you can have staging database tables for data acquisition that continues until all the data is obtained before you populate the actual database tables. This is a lot of extra work.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:04

community wiki
JeeBee

3

To a database, null translates to "I don't have a value for this". Which means that (interestingly), a boolean column that allows nulls is perfectly acceptable, and appears in many database schemas. In contrast, if you have a boolean in your code that can have a value of 'true', 'false' or 'undefined', you're likely to see your code wind up on thedailywtf sooner or later :)

So yes, if you need to allow for the possibility of a field not having any value at all, then allowing nulls on the column is perfectly acceptable. It's significantly better than the potential alternatives (empty strings, zero, etc)

Share Improve this answer Follow

answered Oct 2, 2008 at 17:06

community wiki
Dan

I would use a Boolean object for that case.
– James A. N. Stauffer Oct 2, 2008 at 17:24

3

Nulls can be hard to work with, but they make sense in some cases.

Suppose you have an invoice table with a column "PaidDate" which has a date value. What do you put in that column before the invoice has been paid (assuming you don't know beforehand when it will be paid)? It can't be an empty string, because that's not a valid date. It doesn't make sense to give it an arbitrary date (e.g. 1/1/1900) because that date simply isn't correct. It seems the only reasonable value is NULL, because it does not have a value.

Working with nulls in a database has a few challenges, but databases handle them well. The real problems are when you load nulls from your database into your application code. That's where I've found that things are more difficult. For example, in .NET, a date in a strongly typed dataset (mimicking your DB structure) is a value type and cannot be null. So you have to build workarounds.

Avoid nulls when you can, but don't rule them out because they have valid uses.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:13

community wiki
Jim

I would not have an invoice table with a "PaidDate" column, exactly because of the NULL problem. Instead, I'd have "invoice", "payable", and "receivable" tables, with a foreign key linking invoices with payables. This also solves the problem where an invoice is payed in multiple installments. – benjismith Oct 6, 2008 at 6:03

I'd be happy with a NULL PaidDate, no point adding additional tables if business requirements didn't merit them, but hey it's just an example.. Here's another one: Nullable ExpiryDate column for pages in a content management system. As Jim pointed out, adding an arbitrary date makes no sense. – Nick Apr 22, 2009 at 15:25

3

I think you're confusing Conceptual Data Modeling with Physical Data Modeling.

In CDM's if an object has an optional field, you should subtype the object and create a new object for when that field is not null. That's the theory in CDMs

In the physical world we make all sorts of compromises for the real world. In the real world NULLS are more than fine, they are essential

Share Improve this answer

answered Oct 2, 2008 at 17:29

Follow

community wiki
Mark Brady

3

I agree with many of the answers above and also believe that NULL can be used, where appropriate, in a normalized schema design - particularly where you may wish to avoid using some kind of "magic number" or default value which, in turn, could be misleading!

Ultimately though, I think usage of null needs to be well thought out (rather than by default) to avoid some of the assuptions listed in the answers above, particularly where NULL might **be assumed** to be 'nothing' or 'empty', 'unknown' or the 'value hasn't been entered yet'.

Share Improve this answer

answered Oct 6, 2008 at 4:59

Follow

community wiki
RobS

3

null means no value while 0 doesn't, if you see a 0 you don't know the meaning, if you see a null you know it is a missing value

I think nulls are much clearer, 0 and '' are confusing since they don't clearly show the intent of the value stored

Share Improve this answer Follow

answered Apr 22, 2009 at 15:11

community wiki
SQLMenace

3

Unless you are working with toy databases NULLs are inevitable and in the real world we cannot avoid NULL values.

How can you have first name, middle name, last name for every person? When Middle name and Last name are optional, NULLs are there for you. How you can have Fax, Business phone and Office phone for everybody in the blog list?

NULLS are fine, but you have to handle them properly when retrieving. In SQL server 2008 there is a concept of sparse columns where you can avoid the space taken for NULLs.

Don't confuse NULLs with zero or any other value. People do that and say it is right.

Share Improve this answer Follow

edited Nov 2, 2023 at 8:50

community wiki
2 revs, 2 users 73%
naveen

2

One gotcha if you are using an Oracle database. If you save an empty string to a CHAR type column then Oracle will coerce the value to be NULL without asking. So it can be quite difficult to avoid NULL values in string columns in Oracle.

If you are using NULL values, learn to use the SQL command COALESCE, especially with string values. You can then prevent NULL values propogating into your programming language. For example, imagine a person having a FirstName, MiddleName and FamilyName but you want to return a single field;

```
  SELECT FullName = COALESCE(FirstName + ' ', '')
+ COALESCE(MiddleName+ ' ', '') +
COALESCE(FamilyName, '') FROM Person
```

If you don't use COALESCE, if **any** column contains a **NULL** value you get **NULL** returned.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:21

community wiki
Liam Westley

2

Technically, nulls are illegal in relational math on which the relational database is based. So from a purely technical, semantic relational model point of view, no, they are not okay.

In the real world, denormalization and some violations of the model are okay. But, in general, nulls are an indicator that you should look at your overall design more closely.

I am always very wary of nulls and try to normalize them out whenever I can. But that doesn't mean that they aren't the best choice sometimes. But I would definitely lean to the side of "no nulls" unless you are really sure that having the nulls is better in your particular base.

Share Improve this answer Follow

answered Oct 2, 2008 at 19:06

community wiki
Scott Alan Miller

admittedly my relational algebra/calculus is a bit rusty, but i would love to see a reference on the 'nulls are illegal in relational math' assertion... – Steven A. Lowe Oct 6, 2008 at 5:31

Nulls are not "illegal", but they are unnecessary because the resulting ternary logic can be reduced to single-value logic. Admittedly, "can be reduced to" is not "is easily replaced by". – Dour High Arch Nov 20, 2009 at 0:07

2

NULL rocks. If it wasn't necessary in some cases, SQL would not have IS NULL and IS NOT NULL as special-case operators. NULL is the root of the conceptual universal, all else is NOT NULL. Use NULLs freely, whenever it may be possible for a data value to be absent

but not missed. Default values can only compensate for NULL if they are absolutely correct all of the time. For example, if i have a single-bit field "IsReady" it may make perfect sense for this field to have a default value of false and NULL not be allowed, but this implicitly asserts that we *know* that the whatever is not ready, when in fact we may have no such knowledge. Chances are, in a workflow scenario, the person who decides ready-or-not just hasn't had the chance to enter their opinion yet, so a default of false could actually be dangerous, leading them to overlook a decision that appears to have been made but was in fact only defaulted.

as an aside, and in reference to the middle-initial example, my father had no middle name, therefore his middle initial would be NULL - not blank, space, or asterisk - except in the Army where his middle initial was NMI = No Middle Initial. How silly was that?

Share Improve this answer Follow

answered Oct 6, 2008 at 5:39

community wiki
Steven A. Lowe

2

While technically NULLs are ok as a field value, they are quite frequently frowned upon. Depending on how data is written to your database, it is possible (and common) to end up with an empty string value in the field as opposed to a NULL. So, any query that has this field as part of the WHERE clause, would need to handle both scenarios which are needless keystrokes.

Share Improve this answer Follow

answered Nov 14, 2008 at 15:32

community wiki
CNote

2

My controversial opinion for the day - the default of allowing NULLs in database columns was probably the worst universally accepted design decision in all of RDBMs land. Every vendor does it, and it's wrong. NULLs are fine in certain, specific, well thought out instances, but the idea that you have to explicitly disallow NULLs for every column makes negligent nullability way more common than it should be.

Share Improve this answer Follow

edited Nov 6, 2023 at 13:30

community wiki
2 revs, 2 users 50%
mattmc3

1

Personally, I think that nulls should only be used when you are using the field as a foreign key to another table, to symbolize that this record doesn't link to anything in the other table. Other than that, I find that null values are actually very troublesome when programming application logic. Because there is no direct representation of a database null in most programming languages for many data types, it ends up creating a lot of application code to deal with the meaning of these null values. When a DB encounters null integer, and tries, for instance, add a value of 1 to it (aka null + 1), the database will return null, as that is how the logic is defined. However, when a programming language tries to add null and 1, it will

usually thrown an exception. So, your code ends up littered with checks of what to do when the value is null, which often just equates to converting to 0 for numbers, empty string for text, and some null date (1900/1/1?) for date fields.

Share Improve this answer Follow

answered Oct 2, 2008 at 17:09

community wiki
Kibbee

1

I think the question comes down to what you interpret a value of NULL to signify. Yes, there are many interpretations for a NULL value, however some of them posted here should never be used. The true meaning of NULL is determined by the context of your application and should never mean more than one thing. For example, one suggestion was that NULL on a date of birth field would indicate the person was still alive. This is dangerous.

In all simplicity, define NULL and stick to it. I use it to mean "the value in this field is unknown at this time". It means that and ONLY that. If you need it to mean something else AS WELL, then you need to re-examine your data model.

Share Improve this answer Follow

answered Oct 6, 2008 at 4:47

community wiki
Jack

0

It all comes down to normalization versus ease of use and performance issues.

If you are going to stick to complete normalization rules you are going to end up writing stuff that looks like:

Select c.id, c.lastname,..... from customer c left join customerphonenumber cpn on c.id = cpn.customerid left join customeraddress ca on c.id = ca.customerid left join customerphonenumber2 cpn2 on c.id = cpn2.customerid etc, etc, etc

Share Improve this answer Follow

answered Oct 2, 2008 at 19:16

community wiki
Kevin

0

> It seems that if null is valid within the context of the attribute, then it should be allowed.

But what does null *mean*? That's the rub. It's "no value", but there's a dozen different reasons there might be no value there, and "null" doesn't give you any clue which one it means in this case. (Not set yet, not applicable to

this instance, not applicable to this type, not known, not knowable, not found, error, program bug, ...)

> This is very common in Java where object references are often null.

There's a school of thought that says [null references there are bad there, too]. Same problem: what does null *mean*?

IIRC, Java has both "null" and "uninitialized" (though no syntax for the latter). So Gosling realized the folly of using "null" for every kind of "no value". But why stop with [just two]?

Share Improve this answer Follow

answered Apr 22, 2009 at 17:23

community wiki
Ken

Null means whatever null is defined as for that attribute. I could for example define a null middle name as no middle name. But the meaning of null has to be defined. It's the same as any other value. Using the "what does it mean" argument, any value is flawed. If I see an int field, well what does 3 mean? Well you check the documentation and see what the encoding is. – Steve Kuo Apr 22, 2009 at 18:21

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.