# QGIS Application - Feature request #4551
# Expression parser: automatic conversion to string should be consistent

2011-11-17 05:56 PM - Alister Hood

| | | | |
|---|---|---|---|
| **Status:** | Closed | | |
| **Priority:** | Low | | |
| **Assignee:** | | | |
| **Category:** | Vectors | | |
| **Pull Request or Patch supplied:** | No | **Resolution:** | fixed |
| **Easy fix?:** | No | **Copied to github as #:** | 14465 |

**Description**

In the following expression the $AREA is automatically converted into a string:
'AREA = '||$AREA||' m²'
Users will therefore find it confusing that this expression is "invalid" (they would need to put the maths inside a tostring() to make it valid):
'AREA = '||$AREA/10000||' ha'

Would it be possible to automatically convert part of an expression to a string if the rest of the expression requires it to be one?

### Associated revisions

**Revision 3599700d - 2012-06-22 08:16 PM - Jürgen Fischer**

[FEATURE] implement coalesce and concat function expressions (implements #4551)

### History

**#1 - 2011-12-16 02:09 PM - Giovanni Manghi**

*- Target version set to Version 1.7.4*

**#2 - 2012-01-25 11:53 PM - Alister Hood**

(perhaps this should be a separate ticket...)

Also,
If a NULL value is concatenated, the result of the expression is currently null. For example:

I have a layer of property boundaries for my street (well, a friend's street :) )
There are attribute fields which contain:
- the house number of each property in the street.
- a list of the people living at that address (if somebody does currently live there, and we have already recorded their name(s) ).
I want to label each property with both fields. Using this expression:

```
 "HOUSENUM"  || '
'||  "Occupants"
```

, for properties where the "Occupants" field is empty, there is no label.

I think this behaviour would surprise most users, who would expect the property to still be labelled with the house number. I don't know if the expected behaviour would be consistent with normal SQL, but it would certainly be consistent with the principle of least astonishment ;)

**#3 - 2012-01-26 12:06 AM - Jürgen Fischer**

Alister Hood wrote:

> *I don't know if the expected behaviour would be consistent with normal SQL, but it would certainly be consistent with the principle of least astonishment ;)*

It is.  Any operation on NULL (except IS NULL / IS NOT NULL) is NULL.  So you could use CASE WHEN foo IS NULL THEN '' ELSE foo END if you happen to have NULL values in foo.

#### #4 - 2012-01-26 12:38 AM - Alister Hood

Jürgen Fischer wrote:

> *Alister Hood wrote:*
>
> > *I don't know if the expected behaviour would be consistent with normal SQL, but it would certainly be consistent with the principle of least astonishment ;)*
>
> *It is.*

If the SQL compatibility must stay, I guess we could at least add a warning to the help text for ||.

> *Any operation on NULL (except IS NULL / IS NOT NULL) is NULL.  So you could use CASE WHEN foo IS NULL THEN '' ELSE foo END if you happen to have NULL values in foo.*

What do you think of the idea of creating a new operation for that, so instead of using a long CASE statement on every field you could just use NEW_OPERATION_NAME(foo)?  I can imagine this would make some expressions a lot shorter, and easier for people to decipher.

#### #5 - 2012-01-26 12:58 AM - Jürgen Fischer

Alister Hood wrote:

> *What do you think of the idea of creating a new operation for that, so instead of using a long CASE statement on every field you could just use NEW_OPERATION_NAME(foo)?  I can imagine this would make some expressions a lot shorter, and easier for people to decipher.*

we could call it coalesce

#### #6 - 2012-01-26 05:47 AM - Martin Dobias

I agree with Jürgen. The semantic is the same in other SQL databases - both for the operator precedence and NULL handling. People sometimes misunderstand the NULL value - it does not mean "nothing" and it is definitely not a synonym for a zero or an empty string. NULL value has the meaning of "missing data" and therefore must be handled specially. The coalesce function would be the right way to go.

#### #7 - 2012-01-26 10:12 PM - Alister Hood

Yes, that would be a lot easier than building (or reading) a whole lot of CASE statements.

But I guess the most useful for many labelling use cases would be a concat() function like PostgreSQL:
http://www.postgresql.org/docs/9.1/static/functions-string.html#FUNCTIONS-STRING-OTHER

#### #8 - 2012-01-26 11:18 PM - Alister Hood

Ah, the "More info on expression error" is helpful now :)

For the original feature description above, I see that adding some brackets works (now?):
'AREA = '||($AREA/10000)||' ha'

I'm pretty sure I tried that at the time, but maybe I'm mistaken...

I've found a couple of places listing SQL operator precedence, which indicate that / is higher precedence than ||, so the current behaviour is correct. I guess operator precedence is something that should really be covered in the help, or at least the manual. It doesn't really fit into the the current approach of separate help for each operator.

#### #9 - 2012-04-15 10:09 AM - Giovanni Manghi

*- Target version changed from Version 1.7.4 to Version 2.0.0*

#### #10 - 2012-06-22 11:20 AM - Jürgen Fischer

*- Resolution set to fixed*
*- Status changed from Open to Closed*

concat and coalesce (both with arbitrary number of arguments) implemented in commit:3599700d