# Filtering by numeric custom field types broken after update to master

**I updated my redmine installation to solve bug #5778. After doing so, I can no longer filter by custom fields with numeric types.**

When attempting to do so, I get log/postgresql errors in my logs (included below) which indicate that the custom_values.value field is not being properly cast into an integer. I've seen others where it looks like the query is casting the field, but is not filtering out invalid entries. For example, casting an empty string to a float. Though I saw that one before, I haven't reproduced it yet to include here along with the prior example.

## Versions:

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruby</td>
<td>1.8.7 (x86_64-linux)</td>
</tr>
<tr>
<td>RubyGems</td>
<td>1.4.2</td>
</tr>
<tr>
<td>Rack</td>
<td>1.1.2</td>
</tr>
<tr>
<td>Rails</td>
<td>2.3.14</td>
</tr>
<tr>
<td>Active Record</td>
<td>2.3.14</td>
</tr>
<tr>
<td>Active Resource</td>
<td>2.3.14</td>
</tr>
<tr>
<td>Action Mailer</td>
<td>2.3.14</td>
</tr>
<tr>
<td>Active Support</td>
<td>2.3.14</td>
</tr>
<tr>
<td>Application root</td>
<td>/home/redmine/releases/20111204234554</td>
</tr>
<tr>
<td>Environment</td>
<td>production</td>
</tr>
<tr>
<td>Database adapter</td>
<td>postgresql</td>
</tr>
<tr>
<td>Database schema version</td>
<td>20111201201315</td>
</tr>
</tbody>
</table>

## About your Redmine plugins

- Redmine Workflow Viz plugin: 0.0.1
- Redmine Backlogs: master branch (unstable)
- Redmine Tags: 0.0.1

```sql
SELECT * FROM "custom_fields" WHERE (is_for_all='t') AND ( ("custom_fields"."type" = 'IssueCustomField' ) ) ORDER BY position
```

```sql
SELECT count(*) AS count_all FROM "projects" WHERE (((projects.status = 1) AND (projects.id != 97)) AND (projects."lft" >= 98 AND projects."rgt" <= 99))
```

```
LINE 1: ...lues.custom_field_id=30 WHERE custom_values.value = 1913) AN...
```

```
^ HINT: No operator matches the given name and argument type(s). You might need to add explicit type casts.
```

2020-06-14 1/4
(SELECT issues.id FROM issues LEFT OUTER JOIN custom_values ON custom_values.customized_type='Issue' AND
  custom_values.customized_id=issues.id AND custom_values.custom_field_id=30 WHERE custom_values.value = 1913) AND
  (issue_statuses.is_closed='f') AND projects.id = 97) AND (projects.status=1 AND projects.id IN (SELECT em.project_id FROM
  enabled_modules em WHERE em.name='issue_tracking'))
Query::StatementInvalid: PGError: ERROR: operator does not exist: text = integer
  LINE 1: ...lues.custom_field_id=30 WHERE custom_values.value = 1913) AN...
^  
  HINT: No operator matches the given name and argument type(s). You might need to add explicit type casts.
  
  SELECT count(DISTINCT "issues".id) AS count_all FROM "issues" LEFT OUTER JOIN "projects" ON "projects".id = "issues".project_id
  LEFT OUTER JOIN "issue_statuses" ON "issue_statuses".id = "issues".status_id WHERE (issues.id IN (SELECT issues.id FROM issues
  LEFT OUTER JOIN custom_values ON custom_values.customized_type='Issue' AND custom_values.customized_id=issues.id AND
  custom_values.custom_field_id=1 WHERE CAST(custom_values.value AS decimal(60,3)) <= 100.0) AND projects.id = 97) AND
  (projects.status=1 AND projects.id IN (SELECT em.project_id FROM enabled_modules em WHERE em.name='issue_tracking'))

Associated revisions
Revision 8098 - 2011-12-05 21:45 - Jean-Philippe Lang

Fixed: error when filtering by numeric custom field with postgresql (#9719).

History

#1 - 2011-12-05 09:59 - Etienne Massip
- Status changed from New to Confirmed
- Target version set to 1.3.0

Your error triggers with the "equals" operator.

Using the "<=" operator triggers a different error:

Processing IssuesController#index (for 10.132.21.138 at 2011-12-05 09:51:59) [GET]
  Parameters: ["v":["cf_1">["100"], "op">["cf_1">=""], "group_by">"", "project_id">"prctwa", "set_filter">"1", "c">["tracker", "status",
  "priority", "subject", "assigned_to", "updated_on", "done_ratio", "cf_6"], "action">"index", "f">["cf_1", "]", "controller">"issues"
Query::StatementInvalid: PGError: ERREUR: syntaxe en entrée invalide pour le type numeric : « »
  SELECT count(DISTINCT "issues".id) AS count_all FROM "issues" LEFT OUTER JOIN "projects" ON "projects".id = "issues".project_id
  LEFT OUTER JOIN "issue_statuses" ON "issue_statuses".id = "issues".status_id WHERE (issues.id IN (SELECT issues.id FROM issues
  LEFT OUTER JOIN custom_values ON custom_values.customized_type='Issue' AND custom_values.customized_id=issues.id AND
  custom_values.custom_field_id=1 WHERE CAST(custom_values.value AS decimal(60,3)) <= 100.0) AND projects.id = 1) AND
  (projects.status=1 AND projects.id IN (SELECT em.project_id FROM enabled_modules em WHERE em.name='issue_tracking'))
Rendering template within layouts/base
Rendering common/error (500)

#2 - 2011-12-05 16:58 - James Kyle

Etienne Massip wrote:
Your error triggers with the "equals" operator.

Using the "<=" operator triggers a different error:

[...]

That's the one. It looks (to me) like the query doesn't account for empty strings, which CAST doesn't know how to convert.

#3 - 2011-12-05 21:50 - Jean-Philippe Lang
- Status changed from Confirmed to Resolved

Quick fix committed in r8098.

I think we should split the custom_values.value column into several typed columns (eg. text_value, int_value, float_value, date_value, bool_value) for next major release.

#4 - 2011-12-05 23:37 - Etienne Massip

Jean-Philippe Lang wrote:

> I think we should split the custom_values.value column into several typed columns (eg. text_value, int_value, float_value, date_value, bool_value) for next major release.

I like the idea of having a unique value column instead of one per type; each time I see some table with one column per type, its content is ugly; why not some xml data which now belongs to standard SQL?

#5 - 2011-12-06 05:42 - James Kyle

Thanks for this incredibly timely fix!

#6 - 2011-12-06 09:47 - Etienne Massip

Etienne Massip wrote:

> Jean-Philippe Lang wrote:

> I think we should split the custom_values.value column into several typed columns (eg. text_value, int_value, float_value, date_value, bool_value) for next major release.

> I like the idea of having a unique value column instead of one per type; each time I see some table with one column per type, its content is ugly; why not some xml data which now belongs to standard SQL?

XML seems like a odd idea and is not supported by SQLite3 anyway.

Being used to it for a long time, I don't like the idea of having a column per type because of the dirty contents, but I must admit it would solve both issues:

- performance because of in-query transformation of value
- query crash because of wrong value type stored in value column (but this could also be worked out with an additional 'value type' column)
The second issue will come back soon as a defect since it happens when a custom field used to store literals is switched from any type to Number.

#7 - 2011-12-06 16:58 - James Kyle

but this could also be worked out with an additional 'value type' column

This makes the most sense to me. If you add a column for each type any addition of types require changing the database schema in addition to any code associated with displaying, querying, etc. those types. If a 'value_type' column is used, additional types only require modification of relevant code...often just the addition of a case statement.

If a default is provided for unknown types, instead of crashing the application it could render the page but insert something like "unknown type" in the value column. This would seem to require less developer and maintenance resources.

#8 - 2011-12-06 18:53 - Jean-Philippe Lang

Thanks for your feedbacks

James Kyle wrote:

If you add a column for each type any addition of types require changing the database schema in addition to any code associated with displaying, querying, etc. those types.

I didn't mean 1 column per custom field type but 1 column for each data type: integer, float, text, date, bool. These data types can be reused by different custom field types.

#9 - 2011-12-06 19:04 - Etienne Massip

Jean-Philippe Lang wrote:

I didn't mean 1 column per custom field type but 1 column for each data type: integer, float, text, date, bool. These data types can be reused by different custom field types.

We could use the same column for numerics so converting a CF from float to integer is not destructive.

#10 - 2011-12-07 22:41 - Jean-Philippe Lang

- Status changed from Resolved to Closed
- Resolution set to Fixed