redmineorg-copy202205 - Vote #71012

Filtering by numeric custom field types broken after update to master

2022/05/09 15:55 - Admin Redmine

ステータス:	Closed	**開始日:**	2022/05/09
優先度:	通常	**期日:**	
担当者:		**進捗率:**	0%
カテゴリ:	Custom fields_14	**予定工数:**	0.00時間
対象バージョン:	1.3.0_36	**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/9719	**status_id:**	5
category_id:	14	**tracker_id:**	1
version_id:	36	**plus1:**	0
issue_org_id:	9719	**affected_version:**	
author_id:	45595	**closed_on:**	
assigned_to_id:	0	**affected_version_id:**	
comments:	10		

説明

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 property 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:

```
Ruby version                                1.8.7 (x86_64-linux)
RubyGems version                            1.4.2
Rack version                                1.1.2
Rails version                               2.3.14
Active Record version                       2.3.14
Active Resource version                     2.3.14
Action Mailer version                       2.3.14
Active Support version                      2.3.14
Application root                            /home/redmine/releases/20111204234554
Environment                                 production
Database adapter                            postgresql
Database schema version                     20111201201315

About your Redmine plugins
Redmine Workflow Viz plugin                 0.0.1
Redmine Backlogs                            master branch (unstable)
Redmine Tags                                0.0.1

  IssueCustomField Load (1.2ms)       SELECT * FROM "custom_fields" WHERE (is_for_all='t') AND ( ("custom_fields"."type"
ustomField' ) ) ORDER BY position
    IssueCustomField Load (1.4ms)       SELECT * FROM "custom_fields" INNER JOIN "custom_fields_projects" ON "custom
= "custom_fields_projects".custom_field_id WHERE ("custom_fields_projects".project_id = 97 ) AND ( ("custom_fields"."type" =
ueCustomField' ) ) ORDER BY custom_fields.position
    SQL (1.5ms)       SELECT count(*) AS count_all FROM "projects" WHERE (((projects.status = 1) AND (projects.id !=
ects."lft" >= 98 AND projects."rgt" <= 99))
    SQL (0.0ms)       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 = "iss
_id LEFT OUTER JOIN "issue_statuses" ON "issue_statuses".id = "issues".status_id WHERE (issues.id IN (SELECT issues.id
ssues 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
= 97) AND (projects.status=1 AND projects.id IN (SELECT em.project_id FROM enabled_modules em WHERE em.name='
cking'))
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 = "
t_id  LEFT OUTER JOIN "issue_statuses" ON "issue_statuses".id = "issues".status_id WHERE (issues.id IN (SELECT issue
  issues LEFT OUTER JOIN custom_values ON custom_values.customized_type='Issue' AND custom_values.customized_id=issue
  AND custom_values.custom_field_id=30 WHERE custom_values.value = 1913) AND (issue_statuses.is_closed='f') AND projec
  = 97) AND (projects.status=1 AND projects.id IN (SELECT em.project_id FROM enabled_modules em WHERE em.name
acking'))
Rendering template within layouts/base
Rendering common/error (500)
```

journals

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"=>["tracke
, "status", "priority", "subject", "assigned_to", "updated_on", "done_ratio", "cf_6"], "action"=>"index", "f"=>["cf_1", ""], "controll
ssues"}
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 = "
t_id  LEFT OUTER JOIN "issue_statuses" ON "issue_statuses".id = "issues".status_id WHERE (issues.id IN (SELECT issue
  issues LEFT OUTER JOIN custom_values ON custom_values.customized_type='Issue' AND custom_values.customized_id=issue
  AND custom_values.custom_field_id=1 WHERE CAST(custom_values.value AS decimal(60,3)) <= 100.0) AND projects.id =
D (projects.status=1 AND projects.id IN (SELECT em.project_id FROM enabled_modules em WHERE em.name='issue_trackin
Rendering template within layouts/base
Rendering common/error (500)
```

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.

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.

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?

Thanks for this incredibly timely fix!

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.

> 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.

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.

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.

履歴

#1 - 2022/05/10 17:17 - Admin Redmine

- カテゴリ を Custom fields_14 にセット

- 対象バージョン を 1.3.0_36 にセット