

Redmine - Patch #29171

Add an index to improve the performance of issue queries involving custom fields

2018-07-02 19:39 - Stephane Evr

Status:	Closed	Start date:	
Priority:	Normal	Due date:	
Assignee:	Go MAEDA	% Done:	0%
Category:	Performance	Estimated time:	0.00 hour
Target version:	5.1.0		
Description			
<p>On our Redmine installation, we have around 100000+ issues with lots of custom fields. I ran into a bottleneck were some Issue Queries where very very slow when requesting criteria on multiple custom fields and grouping.</p> <p>In the custom_values table, I noticed that there was no index for [customized_type, customized_id, custom_field_id]. Adding such index resulted in loading times much faster for those complex issue queries (From 60+ seconds down to 5 seconds, with DB caching disabled).</p> <p>Here is the index I added:</p> <pre>class AddMissingIndexCustomValues < ActiveRecord::Migration def change add_index :custom_values, [:customized_type, :customized_id, :custom_field_id], name: "custom_values_customized_custom_field" end end</pre>			

Associated revisions

Revision 21955 - 2022-11-11 00:49 - Go MAEDA

Add an index to improve the performance of issue queries involving custom fields (#29171).

Contributed by Stephane Evr.

History

#1 - 2018-07-02 21:46 - Pavel Rosický

there're already two indexes

```
custom_values_customized [customized_type, customized_id]
index_custom_values_on_custom_field_id [custom_field_id]
```

it would be better to replace custom_values_customized with [customized_type, customized_id, custom_field_id]. We don't need an extra index and I'll still be effective.

60s to 5s seems to be a lot. My db has 140000 issues and 10000000 custom values (mysql 5.7) and I can confirm grouping queries with custom fields are about 50% faster (without db caching). It's an improvement.

What is your db backend? Could you share query plans (explain) of the problematic query (with and without the index)?

<https://dev.mysql.com/doc/refman/8.0/en/explain-extended.html>

#2 - 2018-07-03 16:16 - Stephane Evr

Here is my DB Version: mysql Ver 15.1 Distrib 10.0.34-MariaDB, for debian-linux-gnu (x86_64) using readline 5.2

Please find below an example of long SQL Query. The original query was much much bigger, but I have isolated a part which was taking a lot of time. For instance, sort the list of issues by a Custom Field:

```
SELECT issues.* FROM issues
LEFT OUTER JOIN custom_values cf_34
  ON cf_34.customized_type = 'Issue'
  AND cf_34.customized_id = issues.id
  AND cf_34.custom_field_id = 34
  AND cf_34.value <> ''
ORDER BY Coalesce(cf_34.value, '') DESC
```

LIMIT 25;

Here Custom Field id is 34, it is of type list in Redmine and its possible values are ['OK', 'KO', '']

The DB contains:

- 147262 Issues
- 3318197 Custom Values
- 51211 Custom Values associated with Custom Field 34

Running the query without the added index Takes 13 seconds:

...
25 rows in set (13.64 sec)

And the EXPLAIN:

```
***** 1. row *****
  id: 1
  select_type: SIMPLE
  table: issues
  type: ALL
possible_keys: NULL
  key: NULL
  key_len: NULL
  ref: NULL
  rows: 147262
  Extra: Using temporary; Using filesort
***** 2. row *****
  id: 1
  select_type: SIMPLE
  table: cf_34
  type: ref
possible_keys: custom_values_customized,index_custom_values_on_custom_field_id
  key: custom_values_customized
  key_len: 96
  ref: const,redmine_development.issues.id
  rows: 14
  Extra: Using where
```

Now with the added index, it takes 2 seconds:

...
25 rows in set (1.91 sec)

And the EXPLAIN:

```
***** 1. row *****
  id: 1
  select_type: SIMPLE
  table: issues
  type: ALL
possible_keys: NULL
  key: NULL
  key_len: NULL
  ref: NULL
  rows: 147262
  Extra: Using temporary; Using filesort
***** 2. row *****
  id: 1
  select_type: SIMPLE
  table: cf_34
  type: ref
possible_keys: custom_values_customized,index_custom_values_on_custom_field_id,custom_values_customized_custom_field
  key: custom_values_customized_custom_field
  key_len: 100
  ref: const,redmine_development.issues.id,const
  rows: 1
  Extra: Using where
```

#3 - 2018-07-03 16:24 - Stephane Evr

Pavel Rosický wrote:

it would be better to replace custom_values_customized with [customized_type, customized_id, custom_field_id]. We don't need an extra index and I'll still be effective.

I agree we could just replace the existing index with the new one, though I don't know if this may slow down things somewhere else.

#4 - 2018-07-03 17:17 - Pavel Rosický

ok, the real problem is elsewhere

```
SELECT issues.* FROM issues
LEFT OUTER JOIN custom_values cf_34
  ON cf_34.customized_type = 'Issue'
  AND cf_34.customized_id = issues.id
  AND cf_34.custom_field_id = 34
  AND cf_34.value <> ''
ORDER BY Coalesce(cf_34.value, '') DESC
LIMIT 25;
```

```
id: 1
select_type: SIMPLE
table: issues
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 147262
Extra: Using temporary; Using filesort
```

it means that your db has to load 147262 issues to execute your query. The query is also ordered

```
ORDER BY Coalesce(cf_34.value, '') DESC
```

because it's ordered by a join statement it has to process

14 * 147262 rows

if you add an index the locality is better, then we have to process only

1 * 147262 rows

Maybe we can use inner join for IS/ALL filters (give it a try), but it would require major changes how redmine stores custom values right now. There always has to be a custom value and it also won't work for NULL values. Let's discuss about it in a new ticket if you're interested.

```
SELECT issues.* FROM issues
INNER JOIN custom_values cf_34
  ON cf_34.customized_type = 'Issue'
  AND cf_34.customized_id = issues.id
  AND cf_34.custom_field_id = 34
  AND cf_34.value <> ''
ORDER BY Coalesce(cf_34.value, '') DESC
LIMIT 25;
```

Stephane Evr wrote:

Pavel Rosický wrote:

it would be better to replace custom_values_customized with [customized_type, customized_id, custom_field_id]. We don't need an extra index and I'll still be effective.

I agree we could just replace the existing index with the new one, though I don't know if this may slow down things somewhere else.

writes could be slower because [customized_type, customized_id, custom_field_id] is more complicated than [customized_type, customized_id] it won't slowdown existing read queries because the order is the same as the previous index, for instance

```
SELECT custom_values WHERE customized_type = 'Issue' AND customized_id = 1 can use [customized_type, customize
d_id, custom_field_id] index
```

but

```
SELECT custom_values WHERE customized_id = 1 AND custom_field_id = 1 can't (not a real case)
```

#5 - 2018-07-03 18:09 - Stephane Evr

Pavel Rosický wrote:

because it's ordered by a join statement it has to process
14 * 147262 rows
if you add an index the locality is better, then we have to process only
1 * 147262 rows

Maybe we can use inner join for IS/ALL filters (give it a try), but it would require major changes how redmine stores custom values right now. There always has to be a custom value and it also won't work for NULL values. Let's discuss about it in a new ticket if you're interested.

Okay, thanks for your comments! One thing I am missing is why 14 rows? I tried with a completely different custom field and there were 14 rows to process as well.

Anyway I will keep this index as it has really decreased the response time of Issue#index by a lot in complex projects. For sure there is a small overhead but only as new issues are created (existing custom values will not be reindexed when their value changes).

#6 - 2018-07-03 22:19 - Pavel Rosický

Stephane Evr wrote:

Okay, thanks for your comments! One thing I am missing is why 14 rows? I tried with a completely different custom field and there were 14 rows to process as well.

It is showing how many rows it ran through to get result (it's just an estimate, not an exact number). It depends on many factors, but if the number of rows is too high the query is probably too complex or indexes are missing.

Anyway I will keep this index as it has really decreased the response time of Issue#index by a lot in complex projects. For sure there is a small overhead but only as new issues are created (existing custom values will not be reindexed when their value changes).

I don't see any downsides about this change, so +1

#7 - 2018-07-04 01:35 - Go MAEDA

- *Category set to Performance*

Stephane Evr and Pavel Rosický, thank you for the detailed investigation.

My understanding is that the conclusion is that Redmine should have an index for [:customized_type, :customized_id, :custom_field_id] instead of [customized_type, customized_id]. Is it correct?

#8 - 2018-07-04 12:37 - Stephane Evr

Go MAEDA wrote:

Stephane Evr and Pavel Rosický, thank you for the detailed investigation.

My understanding is that the conclusion is that Redmine should have an index for [:customized_type, :customized_id, :custom_field_id] instead of [customized_type, customized_id]. Is it correct?

Yes

#9 - 2018-07-04 14:42 - Go MAEDA

- *Target version set to Candidate for next major release*

#10 - 2022-10-12 16:38 - Go MAEDA

- *File 29171.patch added*

- *Target version changed from Candidate for next major release to 5.1.0*

Setting the target version to 5.1.0.

#11 - 2022-11-11 00:51 - Go MAEDA

- *Subject changed from Add missing index to custom_values to Add an index to improve the performance of issue queries involving custom fields*

- *Status changed from New to Closed*

- *Assignee set to Go MAEDA*

Committed the patch. Thank you.

#12 - 2023-09-13 15:40 - Han Boetes

Dear @Go MAEDA

Can this patch be added to 5.0.6, please? 5.1.0 is a long way off and this fix would be very welcome.

Thanks,
Han

#13 - 2023-09-14 09:40 - Go MAEDA

Han Boetes wrote in [#note-12](#):

Can this patch be added to 5.0.6, please? 5.1.0 is a long way off and this fix would be very welcome.

I don't think 5.0.6 should include this change, because minor version upgrades of Redmine customarily do not include database migrations.

#14 - 2024-09-09 16:00 - Thijs Thiessens

Hi!

Is there some way we can apply this manually?

Thanks,

Thijs

Files

29171.patch	696 Bytes	2022-10-12	Go MAEDA
-------------	-----------	------------	----------