Redmine - Defect #37962

Missing where cause for allowed_to_condition

2022-11-24 09:54 - Alexander Meindl

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Performance	**Estimated time:**	0.00 hour
Target version:	Candidate for next major release		
Resolution:		**Affected version:**	5.0.3

Description

A missing cause in a database query is missing, namely if allowed_to_condition checks enabled modules.
In Redmine instances with many projects this leads to massiv performance issues.

The attached patch solves this bug. All tests are running with it (no errors) and I found no side effects when using it.

Here are some statistics with performance tests to analyze what the fix accomplishes:

- Redmine 5.0-stable
- Ruby 3.1.2
- 1.200 Projects
- 52.000 issues
- 225.066 time entries

Issues with default filter (status: open) without patch

```
Completed 200 OK in 5402ms (Views: 291.8ms | ActiveRecord: 5066.4ms | Allocations: 278652)
```

Issues with default filter (status: open) with patch

```
Completed 200 OK in 1228ms (Views: 262.1ms | ActiveRecord: 925.3ms | Allocations: 279716)
```

Time entries without filters and default columns and without patch

```
Completed 200 OK in 858028ms (Views: 354.5ms | ActiveRecord: 856707.6ms | Allocations: 914740)
```

Time entries without filters and default columns and with patch

```
Completed 200 OK in 2879ms (Views: 321.3ms | ActiveRecord: 1543.3ms | Allocations: 888275)
```

It would be great, if this patch could be appled to current stable to solve this massive performance issues.

History

#1 - 2022-11-24 09:58 - Alexander Meindl

Sorry, for time entries I switch test results for with and without patch in issue description :(

PS: I used MySQL 8.0.31 for this tests and objects per page is 50 for lists.

#2 - 2022-11-24 10:45 - Go MAEDA

- Description updated

Alexander Meindl wrote:

> Sorry, for time entries I switch test results for with and without patch in issue description :(

Fixed the description.

#3 - 2022-11-26 06:09 - Omega Code

Hello, I have tried your patch under similar circumstances (mysql 8, number of entities) with Redmine 4.2.8 (identical ruby code for those queries) and

I've got approx 1.5x worse time for Issues and Time entries.. checked it many times over, no idea why is that.

my rendered queries are something like this

Before patching

```
SELECT COUNT(DISTINCT `time_entries`.`id`) FROM `time_entries` INNER JOIN `projects` ON `projects`.`id` = `tim
e_entries`.`project_id` INNER JOIN `users` ON `users`.`id` = `time_entries`.`user_id` AND `users`.`type` IN ('
User') LEFT OUTER JOIN `enumerations` ON `enumerations`.`id` = `time_entries`.`activity_id` AND `enumerations`
.`type` IN ('TimeEntryActivity') LEFT OUTER JOIN issues ON issues.id = time_entries.issue_id AND (projects.sta
tus <> 9 AND EXISTS (SELECT 1 AS one FROM enabled_modules em WHERE em.project_id = projects.id AND em.name='ti
me_tracking')) WHERE (projects.status <> 9 AND EXISTS (SELECT 1 AS one FROM enabled_modules em WHERE em.projec
t_id = projects.id AND em.name='time_tracking'))
```

```
SELECT  `time_entries`.`id` AS t0_r0, `time_entries`.`project_id` AS t0_r1, `time_entries`.`user_id` AS t0_r3,
 `time_entries`.`issue_id` AS t0_r4, `time_entries`.`hours` AS t0_r5, `time_entries`.`comments` AS t0_r6, `tim
e_entries`.`activity_id` AS t0_r7, `time_entries`.`spent_on` AS t0_r8, `time_entries`.`tyear` AS t0_r9, `time_
entries`.`tmonth` AS t0_r10, `time_entries`.`tweek` AS t0_r11, `time_entries`.`created_on` AS t0_r12, `time_en
tries`.`updated_on` AS t0_r13, `time_entries`.`rate_id` AS t0_r14, `time_entries`.`cost` AS t0_r15, `enumerati
ons`.`id` AS t1_r0, `enumerations`.`name` AS t1_r1, `enumerations`.`position` AS t1_r2, `enumerations`.`is_def
ault` AS t1_r3, `enumerations`.`type` AS t1_r4, `enumerations`.`active` AS t1_r5, `enumerations`.`project_id`
 AS t1_r6, `enumerations`.`parent_id` AS t1_r7, `enumerations`.`position_name` AS t1_r8 FROM `time_entries` INN
ER JOIN `projects` ON `projects`.`id` = `time_entries`.`project_id` INNER JOIN `users` ON `users`.`id` = `time
_entries`.`user_id` AND `users`.`type` IN ('User') LEFT OUTER JOIN `enumerations` ON `enumerations`.`id` = `ti
me_entries`.`activity_id` AND `enumerations`.`type` IN ('TimeEntryActivity') LEFT OUTER JOIN issues ON issues.
id = time_entries.issue_id AND (projects.status <> 9 AND EXISTS (SELECT 1 AS one FROM enabled_modules em WHERE
 em.project_id = projects.id AND em.name='time_tracking')) WHERE (projects.status <> 9 AND EXISTS (SELECT 1 AS
 one FROM enabled_modules em WHERE em.project_id = projects.id AND em.name='time_tracking')) ORDER BY time_ent
ries.spent_on DESC, time_entries.created_on DESC, time_entries.id ASC
```

After patching

```
SELECT COUNT(DISTINCT `time_entries`.`id`) FROM `time_entries` INNER JOIN `projects` ON `projects`.`id` = `tim
e_entries`.`project_id` INNER JOIN `users` ON `users`.`id` = `time_entries`.`user_id` AND `users`.`type` IN ('
User') LEFT OUTER JOIN `enumerations` ON `enumerations`.`id` = `time_entries`.`activity_id` AND `enumerations`
.`type` IN ('TimeEntryActivity') LEFT OUTER JOIN issues ON issues.id = time_entries.issue_id AND (projects.sta
tus <> 9 AND EXISTS (SELECT 1 AS one FROM enabled_modules em WHERE em.project_id = projects.id AND em.name='ti
me_tracking' AND em.project_id=time_entries.project_id)) WHERE (projects.status <> 9 AND EXISTS (SELECT 1 AS o
ne FROM enabled_modules em WHERE em.project_id = projects.id AND em.name='time_tracking' AND em.project_id=tim
e_entries.project_id))
```

```
SELECT  `time_entries`.`id` AS t0_r0, `time_entries`.`project_id` AS t0_r1, `time_entries`.`user_id` AS t0_r3,
 `time_entries`.`issue_id` AS t0_r4, `time_entries`.`hours` AS t0_r5, `time_entries`.`comments` AS t0_r6, `tim
e_entries`.`activity_id` AS t0_r7, `time_entries`.`spent_on` AS t0_r8, `time_entries`.`tyear` AS t0_r9, `time_
entries`.`tmonth` AS t0_r10, `time_entries`.`tweek` AS t0_r11, `time_entries`.`created_on` AS t0_r12, `time_en
tries`.`updated_on` AS t0_r13, `time_entries`.`rate_id` AS t0_r14, `time_entries`.`cost` AS t0_r15, `enumerati
ons`.`id` AS t1_r0, `enumerations`.`name` AS t1_r1, `enumerations`.`position` AS t1_r2, `enumerations`.`is_def
ault` AS t1_r3, `enumerations`.`type` AS t1_r4, `enumerations`.`active` AS t1_r5, `enumerations`.`project_id`
 AS t1_r6, `enumerations`.`parent_id` AS t1_r7, `enumerations`.`position_name` AS t1_r8 FROM `time_entries` INN
ER JOIN `projects` ON `projects`.`id` = `time_entries`.`project_id` INNER JOIN `users` ON `users`.`id` = `time
_entries`.`user_id` AND `users`.`type` IN ('User') LEFT OUTER JOIN `enumerations` ON `enumerations`.`id` = `ti
me_entries`.`activity_id` AND `enumerations`.`type` IN ('TimeEntryActivity') LEFT OUTER JOIN issues ON issues.
id = time_entries.issue_id AND (projects.status <> 9 AND EXISTS (SELECT 1 AS one FROM enabled_modules em WHERE
 em.project_id = projects.id AND em.name='time_tracking' AND em.project_id=issues.project_id)) WHERE (projects
.status <> 9 AND EXISTS (SELECT 1 AS one FROM enabled_modules em WHERE em.project_id = projects.id AND em.name
='time_tracking' AND em.project_id=issues.project_id)) ORDER BY time_entries.spent_on DESC, time_entries.creat
ed_on DESC, time_entries.id ASC
```

#4 - 2022-11-26 07:51 - Alexander Meindl

Hi Omega Code,

this is strange. The patch adds AND em.project_id=issues.project_id or AND em.project_id=time_entries.project_id to the subquery for enabled modules check. Without this additional case, time entries are not usable in my tests (because it's very slow). #33431 improves performance, but for my setups performance is not good enough. #33431 only reduces joins, if tables are not required - but in some cases all tables are required (e.g. if filter or group by - or combinations of it - requires a table).

Only with this patch I could reduce time entries list from 10 minutes to some seconds. Tests with multiple databases and I also checked if all indexes are correct in database.

I thought worst case with this patch is, that someone do not have performance improvements - but I am not sure, how it could be worse with it.

#5 - 2022-11-27 10:56 - Omega Code

I am sending an update because I believe there was some interference with my previous tests duo the fact that I tried (with some degree of success) to solve this long loading spent time issue with indexes on time_entries table. Now I removed it and my results regarding your patch go as follows:

1. Global Spent time view without filtering on project: **tremendous** difference for better, results comparable to what you have stated
2. Spent time view with filtering on project: no noticable difference
3. Issues global view: difference for worse, it doubled load time for me
4. Issues project view: no noticable difference

I ended up using your patch without changing the /app/models/issue.rb, keeping changes on project and time_entries.
At the same time I am using your #33431 patch.

You're doing a great job trying to optimize the core queries, keep it up!

I spent some considerable time to make "fixes" by using indexes but with limited succes so I think that the apporach for now is changing queries in the app itself.

#6 - 2022-11-27 13:45 - Omega Code

This index on time_entries table helps me a lot with any spent time queries under mysql 8 and Redmine 4.2.8:

```
index_time_entries_on_project_id_user_id_activity_id_issue_id:
`project_id`, `user_id`, `activity_id`, `issue_id`
```

btw. after some observations regarding mysql and late migration from Redmine 3.4 (mysql 5.6) to Redmine 4.2 (mysql 8, formally unsupported here) I believe that mysql Query Cache mechanism (removed in mysql 8) covered many problems with Redmine queries peformance. And now some tinkering, new indexes and optimizations are needed to make up for that under mysql 8.

#7 - 2022-11-28 23:47 - Marius BĂLTEANU

- Target version set to Candidate for next major release

#8 - 2022-12-01 09:56 - Alexander Meindl

I've created a plugin, which can be used to generate sample data for performance tests: https://github.com/alexandermeindl/redmine_sample_data

For MySQL this script needed 15h on my laptop. I added this data dump to the plugin - this dump can be used, of some tests do not want to wait 15 hours, too ;) postgresql data are not in repo (I currently generate it, but these takes time, too)

Files

allowed_to_condition.patch	2.76 KB	2022-11-24	Alexander Meindl