Redmine - Patch #23196

Speed up Project.allowed_to_condition

2016-06-29 14:34 - Jan from Planio www.plan.io

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Jan from Planio www.plan.io	**% Done:**	0%
Category:	Performance	**Estimated time:**	0.00 hour
Target version:			

Description

In cases where there are many issues in a project and the Redmine instance has many projects with many enabled modules overall, Project.allowed_to_condition can get painfully slow (>10 sec for a request in our tests on server grade hardware at Planio with large Redmine databases) because the subselect in source:trunk/app/models/project.rb@15586#L184 will get executed for each of the issues found in the project hierarchy.

Since the list of project ids returned by this query will be static for the scope of the call, it makes sense to cache it. In our tests, this results in a significant speedup (milliseconds vs. seconds).

A patch against current trunk is attached.

Related issues:

Related to Redmine - Patch #21608: Project#allowed_to_condition performance	**Closed**

History

#1 - 2016-06-29 15:36 - Jan from Planio www.plan.io

- File deleted (0001-Eliminate-subselect-in-Project.allowed_to_condition-.patch)

#2 - 2016-06-29 15:37 - Jan from Planio www.plan.io

- File 0001-Eliminate-subselect-in-Project.allowed_to_condition-.patch added

...and a quick fix.

#3 - 2016-06-29 18:36 - Jean-Philippe Lang

> ...will get executed for each of the issues found in the project hierarchy.

I don't get it. In which case the subquery would be executed for each issue in the project hierarchy?
Can you give your DBMS and the numbers of projects/issues you're doing your tests with that result in >10 sec responses, I'd really like to reproduce this performance issue before making any changes. A different one was proposed in #21608 and it would be interesting to compare these two solutions VS. the current implementation.

#4 - 2016-06-29 19:19 - Jean-Philippe Lang

- Related to Patch #21608: Project#allowed_to_condition performance added

#5 - 2016-06-29 20:44 - Jan from Planio www.plan.io

- Assignee set to Jan from Planio www.plan.io

In our scenario it's ~350 projects and ~250,000 issues. The slow loading times occur in projects#show while computing @open_issues_by_tracker = Issue.visible.open.where(cond).group(:tracker).count if the user is admin. We're using MySQL.

The patch brings the load time from ~8 seconds down to a few 100ms.

I will try to come up with a simplified data set that can reproduce this.

#6 - 2016-12-26 08:17 - Alexander Meindl

We had the same performance issue and this patch solved the problem for us.

#7 - 2016-12-26 08:30 - Go MAEDA

- Target version changed from Candidate for next minor release to 3.4.0

#8 - 2016-12-26 08:36 - Go MAEDA

- Status changed from Resolved to New

#9 - 2016-12-26 12:31 - Marius BĂLTEANU

I tried the patch too in our instance and the results are impressive.

Some details about our instance:
- MySQL
- 5289 open issues from a total of 49043
- 166 active projects from a total of 209 projects (13 closed and 30 archived).

I made two basic tests in the global issues page:
1. default query that returns only the open issues:
Before the patch: ~ 8 sec
With the patch applied: ~ 1.4 sec

2. a custom query with all open issues, assigned to some members, project is not a specific one and group by project
Before the patch: ~ 13 sec (some days ago I tried to investigate this slow response and I removed all the plugins from our instance and the result was the same).
With the patch applied: ~ 1.6 sec

#10 - 2016-12-26 16:50 - Jan from Planio www.plan.io

Thanks for your feedback!

#11 - 2017-01-12 23:56 - Jean-Philippe Lang

I've made a few tests with MySQL 5.7 vs PostgreSQL 9.3 with a test database:
2 000 projects, 200 000 issues (100 in each project), 1 000 users
I was logged as a non-admin user who is a member of 300 projects.

I got fast response times with both on projects#show but PostgreSQL behaves much better on the cross project issue list: PG ~ 0.4s / MySQL ~ 6.1s.
With the patch applied, I get better response times with MySQL: PG ~ 0.4s (no change) / MySQL ~ 4.2s

The problem is that PostgreSQL gets slower on pagination with the patch applied: page 50 takes 0.4s without the patch (same response time as the first page) but takes 1.6s with the patch applied.

I've also tested the patch proposed in #21608: PG ~ 0.4s (no change) / MySQL ~ 3.3s. And here, the pagination is not affected with PostgreSQL.

So at the and, #21608 seems to behave better, could you guys give it a try too? I don't really intend to commit a patch that would affect PostgreSQL performance.

#12 - 2017-01-14 09:05 - Alexander Meindl

I created a script to create testing data: https://gist.github.com/alexandermeindl/ed6ae8a199f1b4869ca0f581b9d94f16#file-sample_data-rb. On my laptop this take some hours to run.

My first impression is, that everything runs much smother with PostgreSQL (I only had experience with Redmine and MySQL till now). I've to recreate the test data to get meaningful results. Maybe it is a good idea, to run the tests with enabled and disabled caching.

#13 - 2017-01-14 09:50 - Jean-Philippe Lang

Alexander Meindl wrote:

> I created a script to create testing data: https://gist.github.com/alexandermeindl/ed6ae8a199f1b4869ca0f581b9d94f16#file-sample_data-rb. On my laptop this take some hours to run.

Yes, creating thousands of records using models can be slow. I've made a script too, but using some raw insert queries to do it much faster. I will add it the repository soon.

#14 - 2017-01-16 21:00 - Marius BĂLTEANU

I made the same two tests using the patch from #21608 and the results are almost the same with the one obtained with the patch from this ticket.

#15 - 2017-01-21 10:39 - Jean-Philippe Lang

- Status changed from New to Closed

- Target version deleted (3.4.0)

I've committed #21608 for 3.4.0 as it seems to be a better option.
Please reopen if needed.

Files

0001-Eliminate-subselect-in-Project.allowed_to_condition-.patch	1.42 KB	2016-06-29	Jan from Planio www.plan.io