

Redmine - Defect #11904

Projects with many members are very slow in Redmine 2.0.3

2012-09-21 09:47 - Kristoffer Renholm

Status:	Closed	Start date:	
Priority:	Normal	Due date:	
Assignee:	Jean-Philippe Lang	% Done:	0%
Category:	Issues	Estimated time:	0.00 hour
Target version:		Affected version:	2.0.3
Resolution:	Fixed		

Description

When having a project with ~100 members (many of them are reporters) Redmine slows down and new issue and show issue takes up to 6 seconds to load. This seems to be due to that each and every Principal is queried one by one, see log below. I have tried to work out where these queries come from but have not succeeded so far. Any ideas?

```
SQL (320.6ms) SELECT `members`.`id` AS t0_r0, `members`.`user_id` AS t0_r1, `members`.`project_id` AS t0_r2, `members`.`created_on` AS t0_r3, `members`.`mail_notification` AS t0_r4, `users`.`id` AS t1_r0, `users`.`login` AS t1_r1, `users`.`hashed_password` AS t1_r2, `users`.`firstname` AS t1_r3, `users`.`lastname` AS t1_r4, `users`.`mail` AS t1_r5, `users`.`admin` AS t1_r6, `users`.`status` AS t1_r7, `users`.`last_login_on` AS t1_r8, `users`.`language` AS t1_r9, `users`.`auth_source_id` AS t1_r10, `users`.`created_on` AS t1_r11, `users`.`updated_on` AS t1_r12, `users`.`type` AS t1_r13, `users`.`identity_url` AS t1_r14, `users`.`mail_notification` AS t1_r15, `users`.`salt` AS t1_r16, `roles`.`id` AS t2_r0, `roles`.`name` AS t2_r1, `roles`.`position` AS t2_r2, `roles`.`assignable` AS t2_r3, `roles`.`builtin` AS t2_r4, `roles`.`permissions` AS t2_r5, `roles`.`issues_visibility` AS t2_r6 FROM `members` LEFT OUTER JOIN `users` ON `users`.`id` = `members`.`user_id` AND `users`.`type` IN ('User', 'AnonymousUser') LEFT OUTER JOIN `member_roles` ON `member_roles`.`member_id` = `members`.`id` LEFT OUTER JOIN `roles` ON `roles`.`id` = `member_roles`.`role_id` WHERE `members`.`project_id` = 234 AND (users.type='User' AND users.status=1)
Principal Load (1.2ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 3 LIMIT 1
Principal Load (1.4ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 6 LIMIT 1
Principal Load (1.2ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 10 LIMIT 1
Principal Load (1.2ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 11 LIMIT 1
Principal Load (1.5ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 13 LIMIT 1
Principal Load (0.8ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 14 LIMIT 1
Principal Load (1.3ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 15 LIMIT 1
Principal Load (0.8ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 17 LIMIT 1
Principal Load (0.8ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 24 LIMIT 1
Principal Load (1.4ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 25 LIMIT 1
Principal Load (1.4ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 26 LIMIT 1
Principal Load (1.2ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 29 LIMIT 1
Principal Load (1.7ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 30 LIMIT 1
Principal Load (3.1ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 32 LIMIT 1
Principal Load (1.6ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 34 LIMIT 1
Principal Load (2.0ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 35 LIMIT 1
Principal Load (1.6ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 38 LIMIT 1
Principal Load (2.0ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 42 LIMIT 1
Principal Load (2.3ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 47 LIMIT 1
Principal Load (316.4ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 49 LIMIT 1
Principal Load (1.2ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 52 LIMIT 1
Principal Load (1.3ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 54 LIMIT 1
Principal Load (2.1ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 57 LIMIT 1
Principal Load (1.1ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 60 LIMIT 1
Principal Load (0.8ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 64 LIMIT 1
Principal Load (0.7ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 67 LIMIT 1
Principal Load (0.9ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 68 LIMIT 1
...
```

Ruby/Rails information:

About your application's environment

```
Ruby version      1.8.7 (x86_64-linux)
RubyGems version  1.4.2
```

```
Rack version          1.4
Rails version         3.2.6
Active Record version 3.2.6
Action Pack version   3.2.6
Active Resource version 3.2.6
Action Mailer version 3.2.6
Active Support version 3.2.6
Middleware            Rack::Cache, ActionDispatch::Static, Rack::Lock, #<ActiveSupport::Cache:
:Strategy::LocalCache::Middlewa0x7ff2d68f84e8>, Rack::Runtime, Rack::MethodOverride, ActionDispatch
h::RequestId, Rails::Rack::Logger, ActionDispatch::ShowExcepti, ActionDispatch::DebugExceptions, A
ctionDispatch::RemoteIp, ActionDispatch::Callbacks, ActiveRecord::ConnectionAdapters::ConnectManag
ement, ActiveRecord::QueryCache, ActionDispatch::Cookies, ActionDispatch::Session::CookieStore, Ac
tionDispatch::Flash, Actiospatch::ParamsParser, ActionDispatch::Head, Rack::ConditionalGet, Rack::
ETag, ActionDispatch::BestStandardsSupport, OpenIdAuthenttion
Application root      /opt/redmine/releases/upgrade-2.0.3
Environment          production
Database adapter      mysql
Database schema version 20120422150750
```

Database: mysql Ver 14.14 Distrib 5.1.52, for redhat-linux-gnu (x86_64) using readline 5.1

Associated revisions

Revision 10440 - 2012-09-22 11:33 - Jean-Philippe Lang

Avoid to run one SQL query per member on Project#assignable_users (#11904).

Revision 10441 - 2012-09-22 11:48 - Jean-Philippe Lang

Use eager loaded #principal association instead of #user (#11904).

Revision 10442 - 2012-09-22 12:12 - Jean-Philippe Lang

Use eager loaded #principal association instead of #user (#11904).

History

#1 - 2012-09-22 11:34 - Jean-Philippe Lang

- Assignee set to Jean-Philippe Lang

Kristoffer Renholm wrote:

This seems to be due to that each and every Principal is queried one by one

Fixed in [r10440](#). The patch is pretty simple, you can try it out to see if it solves your performance issue.

#2 - 2012-09-22 11:41 - Jean-Philippe Lang

FTR, the "New issue" page in a project with 200 members now loads in ~250ms on my dev machine and Redmine 2.1.

#3 - 2012-09-23 21:21 - Daniel Felix

Jean-Philippe Lang wrote:

FTR, the "New issue" page in a project with 200 members now loads in ~250ms on my dev machine and Redmine 2.1.

I tried it with the current Revision 10463, and it loads much faster! Thanks!

#4 - 2012-09-24 08:22 - Kristoffer Renholm

- Status changed from New to Resolved

Thanks, works much better! Impressed of the fast response time.

#5 - 2012-10-22 15:34 - Kristoffer Renholm

In which upcoming Redmine version is this fix included?

#6 - 2013-04-14 16:37 - Filou Centrinov

It has been implemented in Redmine 2.3: <source:branches/2.3-stable/app/models/project.rb#L31>

Status: Close?

#7 - 2013-04-14 17:10 - Kristoffer Renholm

Yes please.

#8 - 2013-04-15 06:45 - Toshi MARUYAMA

- *Status changed from Resolved to Closed*

- *Resolution set to Fixed*

Thank you for your feedback.