Redmine - Patch #23328

Optimize Project#notified_users to improve issue create/update speed

2016-07-14 11:35 - Victor Campos

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Go MAEDA	**% Done:**	0%
Category:	Performance	**Estimated time:**	0.00 hour
Target version:	6.0.0		

Description

Hi guys,
When Redmine look for what members it should send e-mail, they interate one by one fetching principal.
This is a N + 1 Query problem.

When we have more then 5K users in one project it is a problem. So with a single line change I drop the time for update issue from 5 to 2 seconds.

I hope this help you.

```
Date:   Tue Jul 12 19:37:14 2016 -0300

    improve update/create speed

diff --git a/app/models/project.rb b/app/models/project.rb
index 660a486..88bd8eb 100644
--- a/app/models/project.rb
+++ b/app/models/project.rb
@@ -524,7 +524,7 @@ class Project &lt; ActiveRecord::Base
   # Returns the users that should be notified on project events
   def notified_users
     # TODO: User part should be extracted to User#notify_about?
-    members.select {|m| m.principal.present? && (m.mail_notification? || m.principal.mail_notific
ation == 'all')}.collect {|m| m.principal}
+    members.includes(:principal).select {|m| m.principal.present? && (m.mail_notification? || m.p
rincipal.mail_notification == 'all')}.collect {|m| m.principal}
   end

   # Returns a scope of all custom fields enabled for project issues
```

Associated revisions

Revision 22590 - 2024-01-04 02:23 - Go MAEDA

Optimize Project#notified_users to improve issue create/update speed (#23328).

Patch by Holger Just (@hjust).

History

#1 - 2016-07-14 19:53 - Lucas Arnaud

- File 0001-Improving-performance-of-project-notified_users-by-e.patch added

I resolved this issue a bit different. I changed the **includes** to **eager_load** to explicitly eager load the **principal** association and added a **find_each** to save memory when the quantity of members is to big.

```
members.eager_load(:principal).find_each()
       .select {|m| m.principal.present? && (m.mail_notification? || m.principal.mail_notification == 'all')}
       .collect {|m| m.principal}
```

I've made some tests and these are the results:

# of project members	current method	after patch
6024	6.13s	1.15s
7933	7.57s	1.40s

7935	7.46s	1.32s

#2 - 2016-07-15 15:31 - Victor Campos

Yes, for memory it's a better solution.
=)

Thx for this patch

#3 - 2016-07-16 04:04 - Go MAEDA

- Description updated

#4 - 2016-07-16 04:21 - Go MAEDA

- Status changed from New to Needs feedback

Redmine 3.3.0 uses preload method in Project#notified_users. Please see r15518.
Could you test Redmine 3.3.0?

#5 - 2016-07-16 15:08 - Victor Campos

Go MAEDA wrote:

> Redmine 3.3.0 uses preload method in Project#notified_users. Please see r15518.
> Could you test Redmine 3.3.0?

Hi Go MAEDA,
What is the policy for update redmine stable branch? When 3.3-stable was lunch I update my redmine for it. When I read your comments I realided that there is a lot off new commits, with new features (redmine.lib changed a lot), performance issues fixed, etc.

About this issue, why preload and not eager_load? And I think the Lucas's idea with find_each is good to prevent memory problems.

#6 - 2016-07-17 09:50 - Go MAEDA

- Status changed from Needs feedback to New

- Assignee set to Jean-Philippe Lang

- Target version set to Candidate for next major release

Thanks for the quick feedback.

Victor Campos wrote:

> What is the policy for update redmine stable branch? When 3.3-stable was lunch I update my redmine for it. When I read your comments I realided that there is a lot off new commits, with new features (redmine.lib changed a lot), performance issues fixed, etc.

I am not a commiter, so I can't explain about the policy. But as I know, the branch was used to prepare releasing of 3.3.0. Many revisions were merged from trunk before 3.3.0 is released.

> About this issue, why preload and not eager_load? And I think the Lucas's idea with find_each is good to prevent memory problems.

I would like Jean-Philippe Lang to make a judgment. Setting assignee to Jean-Philippe.

#7 - 2016-08-22 02:49 - Go MAEDA

- Category set to Performance

#8 - 2024-01-02 10:59 - Go MAEDA

- File optinize-project-notified-users.patch added

- Assignee deleted (Jean-Philippe Lang)

I propose a new approach.

The updated Project#notified_user method constructs a subquery that fetches the user IDs directly from the database. The main query then fetches User records where their IDs match those in the subquery.

```
SELECT "users".*
FROM "users"
```

```
WHERE "users"."id" IN (
    SELECT "members"."user_id"
    FROM "members"
    INNER JOIN "users" ON "users"."id" = "members"."user_id"
    WHERE "members"."project_id" = ?
      AND "users"."type" = 'User'
      AND "users"."status" = 1
      AND "users"."id" IS NOT NULL
      AND (
          members.mail_notification = 1
          OR users.mail_notification = 'all'
      )
)
```

The updated method should reduce memory usage and increase performance by avoiding loading unnecessary ActiveRecord objects into memory. And it filters data using SQL instead of Ruby code, which is typically more efficient.

#9 - 2024-01-03 02:19 - Go MAEDA

- File bench-23328.rb added

- File create_test_members.rb added

Below is the result of a benchmark test.

```
$ bin/rake db:fixtures:load
$ bin/rails r create_test_members.rb
$ bin/rails r bench-23328.rb
members.size: 5002

ruby 3.2.2 (2023-03-30 revision e51014f9c0) [arm64-darwin22]
Warming up --------------------------------------
       Redmine 5.1.0     1.000 i/100ms
       Redmine 3.2.6     1.000 i/100ms
       #23328#note-8   960.000 i/100ms
       #23328#note-2     1.000 i/100ms
Calculating -------------------------------------
       Redmine 5.1.0     10.545 (± 9.5%) i/s -     52.000 in   5.009043s
       Redmine 3.2.6    164.284 (± 5.5%) i/s -    819.000 in   5.005648s
       #23328#note-8      9.568k (± 1.7%) i/s -     48.000k in   5.018386s
       #23328#note-2      4.880 (± 0.0%) i/s -     25.000 in   5.144593s

Comparison:
       #23328#note-8:     9567.6 i/s
       Redmine 3.2.6:      164.3 i/s - 58.24x  slower
       Redmine 5.1.0:       10.5 i/s - 907.34x  slower
       #23328#note-2:        4.9 i/s - 1960.41x  slower
```

#10 - 2024-01-03 18:29 - Holger Just

- File bench-23328-9-fixed.rb added

- File bench-23328-10.rb added

Thank you for the patch and the benchmark!

I think your new code is mostly equivalent to the previous one in terms of results. As such, we should be able to change this here without issues. The only difference I can see is that previously, we could theoretically return member groups from Project#notified_users. However, this would likely only be possible with inconsistent data as groups never have their mail_notification column set and should never have a mail_notification flag on the groups project membership). As such, I believe the code is safe.

Unfortunately though, I believe your benchmark is slightly misleading as I found two issues which resulted in disturbed results

- In the #23328#note-8 benchmark, the final query (User.where(id: subquery)) is only built, but never executed in the benchmark. Appending a .to_a at the end fixes this.
- Furthermore, I believe that Rails caches most of the built queries and results during the benchmark and doesn't actually perform most of the subsequent queries for most of the tests. This results in rather distorted results (as e.g. your result in #note-9 shows the version from Redmine 3.2.6 to be 16 times **faster** than what we have now in 5.1.0, which would be quite unexpected)

By not re-using the members list and forcefully reloading the project in each test, I could force Rails to execute the queries during each benchmark iteration. Using my adapted version of your benchmark script in bench-23328-9-fixed.rb, I got rather different results which appear to fall along the expected magnitudes:

```
ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-darwin23]
Warming up --------------------------------------
```

```
       Redmine 5.1.0     1.000 i/100ms
       Redmine 3.2.6     1.000 i/100ms
       #23328#note-8     4.000 i/100ms
       #23328#note-2     1.000 i/100ms
Calculating -------------------------------------
       Redmine 5.1.0      3.522 (± 0.0%) i/s -     18.000 in   5.168409s
       Redmine 3.2.6      0.413 (± 0.0%) i/s -      3.000 in   7.263348s
       #23328#note-8     48.856 (± 4.1%) i/s -    244.000 in   5.002555s
       #23328#note-2      2.863 (± 0.0%) i/s -     15.000 in   5.299347s

Comparison:
       #23328#note-8:       48.9 i/s
       Redmine 5.1.0:        3.5 i/s - 13.87x  slower
       #23328#note-2:        2.9 i/s - 17.07x  slower
       Redmine 3.2.6:        0.4 i/s - 118.28x  slower
```

Here, your version is still much faster than any of the previous versions by a pretty large margin, just slightly smaller than the initial benchmark seemed to indicate :)

With that being said, I'd still propose a slightly different change to leverage the existing infrastructure of the Project.members scope along with some Rails magic to produce an even more efficient SQL query :)

```
diff --git a/app/models/project.rb b/app/models/project.rb
index 082e83f55b..58346d3733 100644
--- a/app/models/project.rb
+++ b/app/models/project.rb
@@ -34,6 +34,7 @@ class Project < ActiveRecord::Base
   # Memberships of active users only
   has_many :members,
            lambda {joins(:principal).where(:users => {:type => 'User', :status => Principal::STATUS_ACTIVE})}
+  has_many :users, through: :members
   has_many :enabled_modules, :dependent => :delete_all
   has_and_belongs_to_many :trackers, lambda {order(:position)}
   has_many :issues, :dependent => :destroy
@@ -625,13 +626,7 @@ def recipients

   # Returns the users that should be notified on project events
   def notified_users
-    # TODO: User part should be extracted to User#notify_about?
-    users =
-      members.preload(:principal).select do |m|
-        m.principal.present? &&
-         (m.mail_notification? || m.principal.mail_notification == 'all')
-      end
-    users.collect {|m| m.principal}
+    users.where('members.mail_notification = ? OR users.mail_notification = ?', true, 'all')
   end

   # Returns a scope of all custom fields enabled for project issues
```

Using MySQL, Redmine will generate the following SQL query:

```
SELECT DISTINCT `users`.*
FROM `users` INNER JOIN `members` ON `members`.`user_id` = `users`.`id`
WHERE
  `users`.`type` = 'User' AND
  `users`.`status` = 1 AND
  (members.project_id = '1') AND
  (members.mail_notification = '1' OR users.mail_notification = 'all')
```

According to my benchmark, this version is a bit faster than your proposed version as it avoids walking over the users table two times (for the inner and outer queries). In bench-23328-10.rb, you find an edited benchmark-script (which relies on the patch in app/models/project.rb). The comparison tests are the same as in my previously edited bench-23328-9-fixed.rb script:

```
ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-darwin23]
Warming up --------------------------------------
       Redmine 5.1.0     1.000 i/100ms
       Redmine 3.2.6     1.000 i/100ms
    has_many_through     6.000 i/100ms
       #23328#note-8     4.000 i/100ms
       #23328#note-2     1.000 i/100ms
Calculating -------------------------------------
       Redmine 5.1.0      3.387 (± 0.0%) i/s -     17.000 in   5.070548s
       Redmine 3.2.6      0.349 (± 0.0%) i/s -      2.000 in   5.729343s
    has_many_through     61.079 (± 8.2%) i/s -    306.000 in   5.034346s
```

```
       #23328#note-8     48.647 (±10.3%) i/s -    244.000 in   5.051855s
       #23328#note-2      2.839 (± 0.0%) i/s -     15.000 in   5.331465s

Comparison:
    has_many_through:        61.1 i/s
       #23328#note-8:        48.6 i/s - 1.26x  slower
       Redmine 5.1.0:         3.4 i/s - 18.03x  slower
       #23328#note-2:         2.8 i/s - 21.51x  slower
       Redmine 3.2.6:         0.3 i/s - 174.93x  slower
```

#11 - 2024-01-04 02:10 - Go MAEDA

- Subject changed from Improve Update/Create issue speed to Optimize Project#notified_users to improve issue create/update speed

- Target version changed from Candidate for next major release to 6.0.0

Thank you for reviewing things in #note-8 and #note-9 and posting a more sophisticated patch!

#12 - 2024-01-04 02:24 - Go MAEDA

- Status changed from New to Closed

- Assignee set to Go MAEDA

Committed the patch in r22590. Thank you for your contribution.

Files

changeset_r619e156986dde1b674fa1e56bad4bc862c6e9df3.diff	904 Bytes	2016-07-14	Victor Campos
0001-Improving-performance-of-project-notified_users-by-e.patch	1.1 KB	2016-07-14	Lucas Arnaud
optimize-project-notified-users.patch	860 Bytes	2024-01-02	Go MAEDA
bench-23328.rb	1.14 KB	2024-01-03	Go MAEDA
create_test_members.rb	329 Bytes	2024-01-03	Go MAEDA
bench-23328-9-fixed.rb	1.36 KB	2024-01-03	Holger Just
bench-23328-10.rb	1.36 KB	2024-01-03	Holger Just