redmineorg-copy202205 - Vote #79783

Deadlock when assigning custom values

2022/05/09 18:42 - Admin Redmine

ステータス:	Closed	**開始日:**	2022/05/09
優先度:	通常	**期日:**	
担当者:		**進捗率:**	0%
カテゴリ:	Performance_53	**予定工数:**	0.00時間
対象バージョン:	4.0.1_145	**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/30465	**status_id:**	5
category_id:	53	**tracker_id:**	3
version_id:	145	**plus1:**	0
issue_org_id:	30465	**affected_version:**	
author_id:	123153	**closed_on:**	
assigned_to_id:	1	**affected_version_id:**	
comments:	2		

説明

There's been some changes in the new Rails version. It looks like deleting records from association is now scope aware.

After upgrading to Redmine 4.0, I've encountered several deadlocks under heavy load. Note that my DB has 30M+ records in custom_values.

```
ActiveRecord::Deadlocked: Mysql2::Error: Deadlock found when trying to get lock; try restarting transaction:
  DELETE FROM `custom_values` WHERE `custom_values`.`id` IN (SELECT `id` FROM
  (SELECT `custom_values`.`id` FROM `custom_values` LEFT OUTER JOIN `custom_fields` ON `custom_fields`.`id` = `custo
ues`.`custom_field_id`
  WHERE `custom_values`.`customized_id` = 134661 AND
  `custom_values`.`customized_type` = 'Issue' AND `custom_values`.`id` IN
  (34100955, 34100961, 34100989, 34100990, 34100992) ORDER BY custom_fields.position) __active_record_temp)
```

the association looks like this

```
has_many :custom_values, lambda {includes(:custom_field).order("#{CustomField.table_name}.position")},
                         :as => :customized,
                         :inverse_of => :customized,
                         :dependent => :delete_all,
                         :validate => false
```

and it's populated at
https://github.com/redmine/redmine/blob/f5daae4041daa5ad6385f7d4cb43655c87515153/lib/plugins/acts_as_customizable/lib/acts_as_customizable.rb#L145

That's pretty nasty Rails magic. If you take a closer look at the query, it's clear that both includes(:custom_field) and order aren't necessary for deleting records (at least in this case).

the problem is that DELETE command locks the database and creating / updating a record like an issue is wrapped in a transaction.
https://github.com/redmine/redmine/blob/f5daae4041daa5ad6385f7d4cb43655c87515153/app/controllers/issues_controller.rb#L552

Ordering records in a subquery causes that many rows have to be locked in order to execute the query.
In my case this query took ~30s to execute under very heavy load. That's bad.

If duration of the whole transaction exceeded innodb_lock_wait_timeout (default value is 50), it fails as 500 internal server error.

After removing the order statement, the same query with the same conditions took ~0.1s and deadlocks were gone.

IMO, order and maybe even include(:custom_field) shouldn't be part of the association, you should always use an additional scope like #sorted if you want to get a sorted result.

The patch is simple and fixes the problem, but I have no idea how to write a test for it. It may also behave differently on other databases and even on mysql it's quite hard to replicate all conditions.

I tested it on

```
Server version: 5.7.20-19-log Percona Server (GPL), Release '19'
```

please consider refactoring this association. Thanks!

journals

Committed, thanks.

履歴

#1 - 2022/05/10 17:03 - Admin Redmine

- カテゴリ を Performance_53 にセット

- 対象バージョン を 4.0.1_145 にセット