redmineorg-copy202205 - Vote #78529

Implementation of visible conditions with inner join instead of subselect

2022/05/09 18:17 - Admin Redmine

ステータス:	Resolved	**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:	Candidate for next major release_32	**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/26122	**status_id:**	3
category_id:	53	**tracker_id:**	3
version_id:	32	**plus1:**	0
issue_org_id:	26122	**affected_version:**	
author_id:	123153	**closed_on:**	
assigned_to_id:	0	**affected_version_id:**	
comments:	9		

説明

The change from #21608 should be reverted because it's a speed regression:

1/ current version

EXISTS (SELECT 1 AS one FROM enabled_modules em WHERE em.project_id = projects.id AND em. NAME = 'issue_

2/ previous version (actually faster even without an index on enabled modules)

projects.id IN (SELECT project_id FROM enabled_modules em WHERE em.project_id = projects.id AND em. NAME = '
ng'

3/ fastest version (in some cases better indexes were used)

INNER JOIN `enabled_modules` ON `enabled_modules`.`project_id` = `projects`.`id` WHERE `enabled_modules`.`name` = 'is
tracking'

or

INNER JOIN `enabled_modules` ON `enabled_modules`.`project_id` = `projects`.`id` AND `enabled_modules`.`name` = 'issue
king'

Patches: * revert of #21608, because the previous version was faster (about 30%) * new index "enabled_modules_name", but it's not very helpful unless you have many projects (3000+) * implementation of visible conditions with inner join instead of subselect, it passes all tests, but it should be refactored. I want to know what do you think about it first

```
Rails version                4.2.8
Ruby version                 2.1.9-p490 (x64-mingw32)
RubyGems version             2.6.12
Rack version                 1.6.8
Middleware                   Rack::Sendfile, Rack::ContentLength, ActionDispatch::Static, Rack::Lock, #, Rack::Runti
verride, ActionDispatch::RequestId, Rails::Rack::Logger, ActionDispatch::ShowExceptions, ActionDispatch::DebugExceptions, Action
Dispatch::RemoteIp, ActionDispatch::Reloader, ActionDispatch::Callbacks, ActiveRecord::ConnectionAdapters::ConnectionManageme
nt, ActiveRecord::QueryCache, ActionDispatch::Cookies, ActionDispatch::Session::CookieStore, ActionDispatch::Flash, ActionDispatc
h::ParamsParser, ActionDispatch::XmlParamsParser, Rack::Head, Rack::ConditionalGet, Rack::ETag, RequestStore::Middleware, Ope
nIdAuthentication
Environment                  development
Database adapter             mysql2
Database schema version      20170607051650
```

journals

I did some additional benchmarking and I was wrong. Exists really performs better.

so the only relevant patch is joins_enabled_modules.patch

Issue.visible is faster on my environment with this patch about 10% compared to exists with all caching disabled. I don't think it's worth to use it on other places.

Edited issue properties based on the feedback and slightly improved the markup of the description.

As a speed regression, I suggest to put this on version#130

Well, it havily depends on what filters and other conditions are used, it isn't easy to measure. For my long time observations using INNER JOINS is slightly faster (especially on big tables like issues) and mysql has better working plans with it.
On the other side, joins can't be passed directly into WHERE statement which is what allowed_to_condition should do - it increases code complexity. I want some opinions if you think it's worth to use it or not.

@Fernando - as discused before, it's not a speed regression just an improvement, so 3.5.0 is fine.

Pavel Rosický wrote:

> Well, it havily depends on what filters and other conditions are used, it isn't easy to measure. For my long time observations using INNER JOINS is slightly faster (especially on big tables like issues) and mysql has better working plans with it.
> On the other side, joins can't be passed directly into WHERE statement which is what allowed_to_condition should do - it increases code complexity. I want some opinions if you think it's worth to use it or not.
> @Fernando - as discused before, it's not a speed regression just an improvement, so 3.5.0 is fine.

I'm removing this from version:"4.1.0" because it is not a simple change and without performance tests that can be run before and after on both mysql and postgres, it is hard to say if the code complexity added by the patch it's worth it or not. Please correct me if I am wrong.

no, I agree. Thanks for the feedback, this can be closed.

履歴

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

- カテゴリ を Performance_53 にセット

- 対象バージョン を Candidate for next major release_32 にセット