Redmine - Defect #10537

Several tables do not have a primary key...indirectly causes problems with PostgreSQL in Rails 2.3

2012-03-24 03:13 - M T

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Database	**Estimated time:**	0.00 hour
Target version:			
Resolution:	Wont fix	**Affected version:**	1.3.1

Description

There are four tables that do not have a primary key defined in redmine:

- custom_fields_trackers
- custom_fields_projects
- groups_users
- project_trackers

This is technically not illegal, but some argue this is not good practice for apps using AR.

My problem is that this exposes what I think is a latent bug in ActiveRecord's postgres adapter in Rails 2.3.14, whereby it can't insert new records in these tables using a generic ActiveRecord:Base class model.

PG environment: 9.1.3 with latest db gem, i.e. pg-0.13.2 for x86.

I am using a generic rails database transfer script to move redmine from a mysql database across to postgres. It fails on these tables. The error message shows that the INSERT statement that AR constructs is putting the clause RETURNING ("ID") onto the INSERT statement, but because there is no column called ID, the insert fails. I have raised a bug report 5562 on (github) rails for this.

Here is the clip of code that does the transfer:

```
          # Now, write out the prod data to the dev db
           DevelopmentModelClass.transaction do
             models.each do |model|
               new_model = DevelopmentModelClass.new(model.attributes)
               if (model.attributes.has_key?('id'))
                 new_model.id = model.id
               end
                 new_model.save(false)
             end
```

Even though this problem is *technically* not a defect in redmine itself, because it looks like we're going to be stuck with rails 2.3 for a while yet, I think this problem needs to be worked around by adding the primary keys. I don't see any harm in it, it's good practice anyway, and it's not a postgres-specific change in terms of redmine db schema.

I have attached a rake db:migrate script to add the missing primary keys.

It also makes it possible to use the very useful "convert" rake script (see attached). Run as db:convert:prod2dev to transfer production to development database, crossing db types in the process. The script was originally written my Rama McIntosh but refined a little by me. (Converting redmine exposed a couple of issues not catered for by the original script).

History

#1 - 2012-03-24 11:05 - Jean-Philippe Lang

- Status changed from New to Closed

- Resolution set to Wont fix

There's no models behind these tables, these are HABTM association tables. They're not supposed to have a primary key.
http://guides.rubyonrails.org/association_basics.html#the-has_and_belongs_to_many-association

#2 - 2012-03-24 22:09 - M T

I accept that the AR modelling paradigm may be not demand it. For me it's purely a pragmatic device just to get the db transfer done. Until whatever this bug in AR-postgres is fixed, the addition of the ID columns appears to be necessary for the purposes of doing the db transfer, after which it can be rolled back off the target. It really does make this type of transfer a lot easier to be able to use a generic script. Unless anyone has any other ideas on this.

#3 - 2012-03-25 21:14 - Jean-Philippe Lang

Milton Taylor wrote:

> I accept that the AR modelling paradigm may be not demand it.

No, this is a requirement. Quoting the Rails guides:

> This table should be created without a primary key [...] That’s required for the association to work properly. If you observe any strange behavior in a has_and_belongs_to_many association like mangled models IDs, or exceptions about conflicting IDs chances are you forgot that bit.

http://guides.rubyonrails.org/association_basics.html#creating-join-tables-for-has_and_belongs_to_many-associations

#4 - 2012-03-25 22:49 - M T

Cool, thanks for the pointers. Interestingly, the fault in the native pg adapter that was giving me the problem referred to above does not seem to exist in the corresponding AR jdbc adapter, but the latter seems to be seriously broken in other respects as I think you have already found. Am presently trying to shine some light on this.

#5 - 2012-03-26 01:38 - laspariseanicko John

- Assignee set to Jean-Baptiste Barth

-

#6 - 2012-03-26 04:29 - kurtenbagabr billaa

-

#7 - 2012-03-26 07:07 - Toshi MARUYAMA

- Assignee deleted (Jean-Baptiste Barth)

#8 - 2012-03-26 07:50 - sheerinjerr aifseng

- Assignee set to Anonymous

-

#9 - 2012-03-26 07:52 - burgsgill aifseng

-

#10 - 2012-03-26 08:59 - Etienne Massip

- Assignee deleted (Anonymous)

#11 - 2012-03-26 09:12 - raankelv John

- Assignee set to M T

-

#12 - 2012-03-26 09:46 - Toshi MARUYAMA

- Assignee deleted (M T)

#13 - 2012-04-03 12:10 - M T

- File convert.rake added

Just as a postscript for this issue, I have further altered the database transfer script to work around the original obstacle I encountered with transferring the HABTM tables. I have used this script to successfully move all redmine data from a mysql database to a postgresql database. It doesn't matter what version of redmine database you do this on - the script is introspective of the db, i.e. it does not use specific lists of tables nor

does it rely on redmine application models. The only requirement is that the two schemas are both at the same rails migration level. As with all these things, make good backups first, and use at your own risk. YMMV.

This rake script really belongs in the wiki somewhere...perhaps one of the dev team might care to create a page for database migration (as in from one db type to another), discussing the various ways this can be accomplished.

Files

20120324000000_add_missing_primary_keys.rb	507 Bytes	2012-03-24	M T
convert.rake	5.51 KB	2012-03-24	M T
convert.rake	6.16 KB	2012-04-03	M T