

## Redmine - Feature #17460

### MySQL 5.7 support

2014-07-15 00:10 - Sam Sheen

<b>Status:</b>	New	<b>Start date:</b>	
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>		<b>% Done:</b>	0%
<b>Category:</b>	Database	<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>	Candidate for next major release		
<b>Resolution:</b>			

#### Description

Env:

OS: Ubuntu 14.04 LTS

ruby 2.1.2p95 (2014-05-08 revision 45877) [i686-linux]

rails (4.1.4, 3.2.19)

redmine-2.5.2

svn, version 1.8.9 (r1591380)

Environment:

Redmine version 2.5.2.stable

Ruby version 2.1.2-p95 (2014-05-08) [i686-linux]

Rails version 3.2.19

Environment production

Database adapter Mysql2

SCM:

Subversion 1.8.9

Filesystem

Redmine plugins:

no plugin installed

After install mysql, redmine, I try to set DB and find a problem in mysql 5.7.

```
root@lenovo:/usr/local/src/redmine-2.5.2# RAILS_ENV=production rake db:migrate
```

```
Setup: migrating =====
```

```
-- create_table("attachments", {:force=>true})
```

```
rake aborted!
```

An error has occurred, all later migrations canceled:

```
Mysql2::Error: All parts of a PRIMARY KEY must be NOT NULL; if you need NULL in a key, use UNIQUE instead: CREATE TABLE `attachments` (`id` int(11) DEFAULT NULL auto_increment PRIMARY KEY, `container_id` int(11) DEFAULT 0 NOT NULL, `container_type` varchar(30) DEFAULT " NOT NULL, `filename` varchar(255) DEFAULT " NOT NULL, `disk_filename` varchar(255) DEFAULT " NOT NULL, `filesize` int(11) DEFAULT 0 NOT NULL, `content_type` varchar(60) DEFAULT "", `digest` varchar(40) DEFAULT " NOT NULL, `downloads` int(11) DEFAULT 0 NOT NULL, `author_id` int(11) DEFAULT 0 NOT NULL, `created_on` datetime) ENGINE=InnoDB
```

```
Tasks: TOP => db:migrate
```

```
(See full trace by running task with --trace)
```

But, redmine is compatible mysql-5.6.19 and work perfectly.

Sam Sheen

#### Related issues:

Related to Redmine - Defect # 19344: MySQL 5.6: IssueNestedSetConcurrencyTest...

**New**

## Associated revisions

---

### Revision 14011 - 2015-02-15 10:09 - Jean-Philippe Lang

Workaround for timestamps rounding issues with Rails4.2 and mysql5.7 that may kill user session after password is changed (#17460).

### Revision 14077 - 2015-03-14 07:31 - Toshi MARUYAMA

add MariaDB 10.0 environment to travis (#17460, #19344)

### Revision 14085 - 2015-03-14 11:16 - Toshi MARUYAMA

add MySQL 5.6 and 5.7 environments to travis (#17460, #19344)

### Revision 14128 - 2015-03-17 00:20 - Toshi MARUYAMA

add MariaDB 5.5 environment to travis (#17460, #19344, #19395)

## History

---

### #1 - 2014-08-03 13:56 - Go MAEDA

This is caused by a change of MySQL 5.7.3-m13. Please see the following URL for details.

[mysql - Creating tables and problems with primary key in Rails - Stack Overflow](#)

The [workaround](#) is included in Rails 4.1. But current Redmine is based on Rails 3.2, so it seems that we have to rely on [monkey patch](#) for now.

### #2 - 2014-08-04 08:18 - Etienne Massip

- Target version set to Candidate for next minor release

Don't know what to do with this one, guess the patch can't be avoided?

### #3 - 2014-11-03 09:35 - Go MAEDA

- Target version deleted (Candidate for next minor release)

Probably this issue will be resolved in Redmine 3.0.0 because it is based on Rails 4.1.

### #4 - 2014-11-03 21:41 - Enderson Maia

Same problem here using redmine-2.6.0.

```
# mysql --version
mysql Ver 5.7.5-m15 for linux-glibc2.5 on x86_64 (MySQL Community Server (GPL))
# bundle exec rails --version
Rails 3.2.19
```

```
# ruby --version
ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-linux-gnu]
```

#### #5 - 2014-11-03 21:47 - Enderson Maia

If you're gonna wait for Rails 4 in Redmine 3, maybe an update to the docs to inform it's not compatible with this specific version of MySQL.

#### #6 - 2014-11-11 10:15 - Jean-Philippe Lang

- Target version set to 3.0.0

Note about this incompatibility added to `[[RedmineInstall]]`.

#### #7 - 2015-02-15 10:12 - Jean-Philippe Lang

- Tracker changed from Defect to Feature

- Subject changed from Redmine 2.5.2 incompatible with mysql-5.7.3-m13 to MySQL 5.7 support

There are still some issues with mysql 5.7 and Rails 4.2:

1. it does not pass the issue concurrency test (dead locks), although the [5.7 changelog](#) does not mention any changes to the lock mechanism
2. timestamps rounding issues after reload that trigger failures in

`AccountTest#test_user_with_must_change_passwd_should_be_able_to_change_its_password`. Here is an example that shows a timestamp returning a different value after reload:

```
irb(main):044:0> u=User.first
irb(main):045:0> u.created_on = "2015-02-15 09:38:59.767393"
=> "2015-02-15 09:38:59.767393"
irb(main):046:0> u.save
=> true
irb(main):047:0> u.created_on
=> Sun, 15 Feb 2015 09:38:59 UTC +00:00
irb(main):048:0> u.reload
irb(main):049:0> u.created_on
=> Sun, 15 Feb 2015 09:39:00 UTC +00:00
```

A workaround was committed in r14011 for 2.

#### #8 - 2015-02-18 13:15 - Jean-Philippe Lang

- Target version changed from 3.0.0 to Candidate for next major release

#### #9 - 2015-03-11 13:02 - Toshi MARUYAMA

Jean-Philippe Lang wrote:

1. it does not pass the issue concurrency test (dead locks), although the [5.7 changelog](#) does not mention

<http://dev.mysql.com/doc/refman/5.6/en/mysql-nutshell.html>

| *InnoDB uses a new, faster algorithm to detect deadlocks.*

**#10 - 2015-03-11 13:04 - Toshi MARUYAMA**

- Related to Defect #19344: MySQL 5.6: IssueNestedSetConcurrencyTest#test\_concurrency : always fails added

**#11 - 2015-03-11 13:07 - Toshi MARUYAMA**

Jean-Philippe Lang wrote:

| 1. *it does not pass the issue concurrency test (dead locks)*

#19344 says MySQL 5.6 too.

**#12 - 2015-03-16 21:20 - Jean-Philippe Lang**

Toshi MARUYAMA wrote:

| *#19344 says MySQL 5.6 too.*

Indeed, the CI server runs MySQL 5.1.

I had a deeper look at the deadlocks issue and it seems to work when doing SELECT \* ... FROM UPDATE instead of SELECT id ... FOR UPDATE. Here is a patch for current trunk tested with mysql5.7, the concurrency test passes for me. Could you give it a try?

**#13 - 2015-03-16 21:20 - Jean-Philippe Lang**

- File *mysql5.7\_deadlocks\_fix.patch* added

**#14 - 2015-03-17 13:45 - Toshi MARUYAMA**

On my CentOS7 mariadb-5.5.41-2.el7\_0.x86\_64:

clean r14128:

```
$ ruby test/unit/issue_nested_set_concurrency_test.rb
```

```
Run options: --seed 12276
```

```
# Running:
```

```
F.
```

```
Finished in 19.053029s, 0.1050 runs/s, 0.4199 assertions/s.
```

```
1) Failure:
```

```
IssueNestedSetConcurrencyTest#test_concurrency [test/unit/issue_nested_set_concurrency_test.rb:45]:
```

```
Expected "Mysql2::Error: Deadlock found when trying to get lock;
```

try restarting transaction:

```
SELECT `issues`.`id` FROM `issues` WHERE (root_id IN (SELECT root_id FROM issues WHERE id IN (319,316)))
ORDER BY `issues`.`id` ASC FOR UPDATE" to be nil.
```

2 runs, 8 assertions, 1 failures, 0 errors, 0 skips

r14128 with note-13 patch:

```
$ ruby test/unit/issue_nested_set_concurrency_test.rb
```

```
Run options: --seed 50424
```

```
# Running:
```

```
F.
```

```
Finished in 5.455071s, 0.3666 runs/s, 1.2832 assertions/s.
```

1) Failure:

```
IssueNestedSetConcurrencyTest#test_concurrency [test/unit/issue_nested_set_concurrency_test.rb:45]:
```

```
Expected "Mysql2::Error: Deadlock found when trying to get lock;
```

```
try restarting transaction:
```

```
SELECT `issues`.* FROM `issues` WHERE `issues`.`root_id` = 432
```

```
ORDER BY `issues`.`id` ASC FOR UPDATE" to be nil.
```

2 runs, 7 assertions, 1 failures, 0 errors, 0 skips

## #15 - 2015-03-17 17:37 - Toshi MARUYAMA

This change passes test half times, but fails half times on my MariaDB 5.5.

```
diff --git a/lib/redmine/nested_set/issue_nested_set.rb b/lib/redmine/nested_set/issue_nested_set.rb
```

```
--- a/lib/redmine/nested_set/issue_nested_set.rb
```

```
+++ b/lib/redmine/nested_set/issue_nested_set.rb
```

```
@@ -158,7 +158,8 @@ module Redmine
```

```
    self.class.reorder(:id).where(:root_id => sets_to_lock).lock(lock).ids
```

```
  else
```

```
    sets_to_lock = [id, parent_id].compact
```

```
-    self.class.reorder(:id).where("root_id IN (SELECT root_id FROM #{self.class.table_name} WHERE id IN (?))", sets_to_lock).lock.ids
```

```
+    root_ids = self.class.where(:id => sets_to_lock).select(:root_id).to_a
```

```
+    self.class.where(:root_id => root_ids).lock.ids
```

```
  end
```

```
end
```

```
$ ruby test/unit/issue_nested_set_concurrency_test.rb
```

```
Run options: --seed 63128
```

# Running:

..

Finished in 25.875842s, 0.0773 runs/s, 0.3865 assertions/s.

2 runs, 10 assertions, 0 failures, 0 errors, 0 skips

\$ ruby test/unit/issue\_nested\_set\_concurrency\_test.rb

Run options: --seed 40861

# Running:

FF

Finished in 6.222392s, 0.3214 runs/s, 0.6428 assertions/s.

1) Failure:

IssueNestedSetConcurrencyTest#test\_concurrency [test/unit/issue\_nested\_set\_concurrency\_test.rb:45]:

Expected "Mysql2::Error: Deadlock found when trying to get lock; try restarting transaction:

UPDATE `issues` SET lft = CASE WHEN lft > 9 THEN lft - 2 ELSE lft END, rgt = CASE WHEN rgt > 9 THEN rgt - 2 ELSE rgt END WHERE `issues`.`root\_id` = 4249 AND (lft > 9 OR rgt > 9)" to be nil.

2) Failure:

IssueNestedSetConcurrencyTest#test\_concurrent\_subtasks\_creation [test/unit/issue\_nested\_set\_concurrency\_test.rb:61]:

Expected "Mysql2::Error: Deadlock found when trying to get lock; try restarting transaction:

UPDATE `issues` SET lft = CASE WHEN lft >= 18 THEN lft + 2 ELSE lft END, rgt = CASE WHEN rgt >= 18 THEN rgt + 2 ELSE rgt END WHERE `issues`.`root\_id` = 4263 AND (lft >= 18 OR rgt >= 18)" to be nil.

2 runs, 4 assertions, 2 failures, 0 errors, 0 skips

#### #16 - 2015-03-17 18:53 - Jean-Philippe Lang

Toshi, your patch does not do what it's supposed to.

You may want to write:

```
root_ids = self.class.where(:id => sets_to_lock).pluck(:root_id)
```

instead of:

```
root_ids = self.class.where(:id => sets_to_lock).select(:root_id).to_a
```

which returns records without their ids. The lock after that does nothing:

```
SELECT `issues`.`id` FROM `issues` WHERE `issues`.`root_id` IS NULL FOR UPDATE
```

#### #17 - 2015-03-17 19:07 - Jean-Philippe Lang

I've isolated the log for a thread that triggers a dead lock. It ends with:

```
[52206168] BEGIN
[52206168] SELECT `issues`.* FROM `issues` WHERE `issues`.`id` = 781 LIMIT 1
[52206168] SELECT `issues`.`id` FROM `issues` WHERE `issues`.`root_id` = 778 ORDER BY `issues`.`id` ASC FOR UPDATE
[52206168] ROLLBACK
[52206168] ERROR: Mysql2::Error: Deadlock found when trying to get lock; try restarting transaction:
SELECT `issues`.`id` FROM `issues` WHERE `issues`.`root_id` = 778 ORDER BY `issues`.`id` ASC FOR UPDATE
```

As we can see, the thread starts a transaction, has no lock yet and gets a dead lock error on the first lock. Anyone knows what would explain that?

#### #18 - 2015-03-17 20:00 - Toshi MARUYAMA

Jean-Philippe Lang wrote:

*Toshi, your patch does not do what it's supposed to.*

*You may want to write:*

```
root_ids = self.class.where(:id => sets_to_lock).pluck(:root_id)
```

*instead of:*

```
root_ids = self.class.where(:id => sets_to_lock).select(:root_id).to_a
```

*which returns records without their ids. The lock after that does nothing:*

```
SELECT `issues`.`id` FROM `issues` WHERE `issues`.`root_id` IS NULL FOR UPDATE
```

This changes fails 3/4 times.

```
diff --git a/lib/redmine/nested_set/issue_nested_set.rb b/lib/redmine/nested_set/issue_nested_set.rb
--- a/lib/redmine/nested_set/issue_nested_set.rb
+++ b/lib/redmine/nested_set/issue_nested_set.rb
@@ -158,7 +158,8 @@ module Redmine
   self.class.reorder(:id).where(:root_id => sets_to_lock).lock(lock).ids
   else
     sets_to_lock = [id, parent_id].compact
-   self.class.reorder(:id).where("root_id IN (SELECT root_id FROM #{self.class.table_name} WHERE id IN (?))", sets_to_lock).lock.ids
+   root_ids = self.class.where(:id => sets_to_lock).pluck(:root_id).compact.uniq
+   self.class.where(:root_id => root_ids).lock.ids
   end
 end
```

```
$ ruby test/unit/issue_nested_set_concurrency_test.rb
```

```
Run options: --seed 1553
```

```
# Running:
```

```
F.
```

Finished in 20.314407s, 0.0985 runs/s, 0.4430 assertions/s.

1) Failure:

IssueNestedSetConcurrencyTest#test\_concurrency [test/unit/issue\_nested\_set\_concurrency\_test.rb:45]:

Expected "Mysql2::Error: Deadlock found when trying to get lock; try restarting transaction: SELECT `issues`.`id` FROM `issues` WHERE `issues`.`root\_id` = 7049 FOR UPDATE" to be nil.

2 runs, 9 assertions, 1 failures, 0 errors, 0 skips

#### #19 - 2015-03-18 05:10 - Toshi MARUYAMA

"SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;" reduces failure times on my MariaDB 5.5.

#### #20 - 2015-03-18 06:29 - Toshi MARUYAMA

This is code from source:tags/2.6.3/lib/plugins/awesome\_nested\_set/lib/awesome\_nested\_set/model/transactable.rb .

```
diff --git a/lib/redmine/nested_set/issue_nested_set.rb b/lib/redmine/nested_set/issue_nested_set.rb
```

```
--- a/lib/redmine/nested_set/issue_nested_set.rb
```

```
+++ b/lib/redmine/nested_set/issue_nested_set.rb
```

```
@@ -148,7 +148,29 @@ module Redmine
```

```
  new_record? || !is_or_is_ancestor_of?(issue)
```

```
  end
```

```
+ def in_tenacious_transaction(&block)
```

```
+   retry_count = 0
```

```
+   begin
```

```
+     transaction(&block)
```

```
+   rescue ActiveRecord::StatementInvalid => error
```

```
+     raise unless error.message =~ /Deadlock found when trying to get lock/
```

```
+     raise unless retry_count < 10
```

```
+     retry_count += 1
```

```
+     logger.info "Deadlock detected on retry #{retry_count}, restarting transaction"
```

```
+     sleep(rand(retry_count)*0.1) # Aloha protocol
```

```
+     retry
```

```
+   end
```

```
+ end
```

```
+
```

```
def lock_nested_set
```

```
+ if self.class.connection.adapter_name =~ /mysql/i
```

```
+   in_tenacious_transaction { lock_nested_set_in_tenacious_transaction }
```

```
+ else
```

```
+   lock_nested_set_in_tenacious_transaction
```

```
+ end
```

```
+ end
```

```
+
```

```
+ def lock_nested_set_in_tenacious_transaction
```

```
  if self.class.connection.adapter_name =~ /sqlserver/i
```

```
    lock = "WITH (ROWLOCK HOLDLOCK UPDLOCK)"
```



```
# Custom lock for SQLServer
```

#### #21 - 2015-03-18 07:07 - Toshi MARUYAMA

Note-20 is wrong because it uses nested transaction and parent transaction does not use lock.

This is fix.

```
diff --git a/lib/redmine/nested_set/issue_nested_set.rb b/lib/redmine/nested_set/issue_nested_set.rb
--- a/lib/redmine/nested_set/issue_nested_set.rb
+++ b/lib/redmine/nested_set/issue_nested_set.rb
@@ -148,7 +148,29 @@ module Redmine
   new_record? || lis_or_is_ancestor_of?(issue)
 end

+ def get_lock_mysql(&block)
+   retry_count = 0
+   begin
+     yield
+   rescue ActiveRecord::StatementInvalid => error
+     raise unless error.message =~ /Deadlock found when trying to get lock/
+     raise unless retry_count < 10
+     retry_count += 1
+     logger.info "Deadlock detected on retry #{retry_count}, restarting transaction"
+     sleep(rand(retry_count)*0.1) # Aloha protocol
+     retry
+   end
+ end
+
+ def lock_nested_set
+   if self.class.connection.adapter_name =~ /mysql/i
+     get_lock_mysql { get_lock }
+   else
+     get_lock
+   end
+ end
+
+ def get_lock
+   if self.class.connection.adapter_name =~ /sqlserver/i
+     lock = "WITH (ROWLOCK HOLDLOCK UPDLOCK)"
+     # Custom lock for SQLServer
```

#### #22 - 2015-03-18 08:23 - Jean-Philippe Lang

Toshi, the idea of the implementation of nested sets in 3.0.0 is to start the transaction by locking all the rows that might be updated or used to compute shifts in the transaction, in order to prevent dead locks and inconsistencies. I won't commit that workaround until I figure out why it doesn't work as I expect (note-17) in recent versions of MySQL.

#### #23 - 2015-03-19 05:00 - Toshi MARUYAMA

I think MySQL uses **Gap Locks**, so we cannot avoid deadlock.

<http://dev.mysql.com/doc/refman/5.6/en/innodb-record-level-locks.html#idm140169015854080>

I tried **READ COMMITTED** on MariaDB 5.5, but deadlock raised. I don't know the reason.

**#24 - 2017-02-01 18:28 - Dave Martin**

Do current versions of Redmine still not support MySQL 5.7?

**#25 - 2017-03-21 16:38 - Toshi MARUYAMA**

#23318#note-18 patch reduces test failure times from about 100% to 50% on my CentOS7 mariadb-5.5.52-1.el7.x86\_64.

**#26 - 2017-04-21 18:11 - Stephane Evr**

Latest versions of Ubuntu server only provide the 5.7 package, version 5.5 is really difficult to install on it:

<https://askubuntu.com/questions/763240/is-it-possible-to-install-mysql-5-5-or-5-6-on-ubuntu-16-04>

**#27 - 2017-04-23 15:45 - Stephane Evr**

Here are some logs from MySQL 5.7:

```
mysql> select * FROM INNODB_LOCKS \G;
***** 1. row *****
lock_id: 56163:265:12:54
lock_trx_id: 56163
lock_mode: X
lock_type: RECORD
lock_table: `redmine_test`.`issues`
lock_index: index_issues_on_root_id_and_lft_and_rgt
lock_space: 265
lock_page: 12
lock_rec: 54
lock_data: 653, 1, 20, 653
***** 2. row *****
lock_id: 56159:265:12:54
lock_trx_id: 56159
lock_mode: X
lock_type: RECORD
lock_table: `redmine_test`.`issues`
lock_index: index_issues_on_root_id_and_lft_and_rgt
lock_space: 265
lock_page: 12
lock_rec: 54
lock_data: 653, 1, 20, 653
2 rows in set, 1 warning (0.00 sec)
```

```
mysql> SHOW ENGINE INNODB STATUS \G;
```

```
-----
```

LATEST DETECTED DEADLOCK

-----

2017-04-23 14:38:12 0x7f308c273700

\*\*\* (1) TRANSACTION:

TRANSACTION 56161, ACTIVE 0 sec starting index read

mysql tables in use 2, locked 1

LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)

MySQL thread id 9, OS thread handle 139846486136576, query id 1081 localhost root Sending data

SELECT `issues`.`id` FROM `issues` WHERE (root\_id IN (SELECT root\_id FROM issues WHERE id IN (658,655))) ORDER BY `issues`.`id` ASC FOR UPDATE

\*\*\* (1) WAITING FOR THIS LOCK TO BE GRANTED:

RECORD LOCKS space id 265 page no 12 n bits 160 index index\_issues\_on\_root\_id\_and\_lft\_and\_rgt of table `redmine\_test`.`issues` trx id 56161 lock\_mode X waiting

Record lock, heap no 47 PHYSICAL RECORD: n\_fields 4; compact format; info bits 32

0: len 4; hex 8000028d; asc ;;

1: len 4; hex 80000001; asc ;;

2: len 4; hex 80000016; asc ;;

3: len 4; hex 8000028d; asc ;;

\*\*\* (2) TRANSACTION:

TRANSACTION 56159, ACTIVE 0 sec updating or deleting

mysql tables in use 1, locked 1

5 lock struct(s), heap size 1136, 60 row lock(s), undo log entries 1

MySQL thread id 7, OS thread handle 139846486537984, query id 1134 localhost root updating

UPDATE `issues` SET lft = CASE WHEN lft > 5 THEN lft - 2 ELSE lft END, rgt = CASE WHEN rgt > 5 THEN rgt - 2 ELSE rgt END WHERE `issues`.`root\_id` = 653 AND (lft > 5 OR rgt > 5)

\*\*\* (2) HOLDS THE LOCK(S):

RECORD LOCKS space id 265 page no 12 n bits 160 index index\_issues\_on\_root\_id\_and\_lft\_and\_rgt of table `redmine\_test`.`issues` trx id 56159 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 47 PHYSICAL RECORD: n\_fields 4; compact format; info bits 32

0: len 4; hex 8000028d; asc ;;

1: len 4; hex 80000001; asc ;;

2: len 4; hex 80000016; asc ;;

3: len 4; hex 8000028d; asc ;;

\*\*\* (2) WAITING FOR THIS LOCK TO BE GRANTED:

RECORD LOCKS space id 265 page no 12 n bits 160 index index\_issues\_on\_root\_id\_and\_lft\_and\_rgt of table `redmine\_test`.`issues` trx id 56159 lock\_mode X locks gap before rec insert intention waiting

Record lock, heap no 47 PHYSICAL RECORD: n\_fields 4; compact format; info bits 32

0: len 4; hex 8000028d; asc ;;

1: len 4; hex 80000001; asc ;;

2: len 4; hex 80000016; asc ;;

3: len 4; hex 8000028d; asc ;;

\*\*\* WE ROLL BACK TRANSACTION (1)

**#28 - 2017-04-24 13:11 - Stephane Evr**

Should we put the index on :issues => [:root\_id, :lft, :rgt] as unique? I think this would play a role in the number of records being locked when we do something such as:

```
In remove_from_nested_set:  
self.class.where(:root_id => root_id).where("lft >= ? AND rgt <= ?", lft, rgt).update_all(...)
```

```
In add_to_nested_set:  
self.class.where(:root_id => root_id).where("lft >= ? OR rgt >= ?", lft, lft).update_all(...)
```

Or is a reorder needed before the update\_all clause ?

**#29 - 2017-05-18 22:22 - Mark Anderson**

Am about to make the move to Ubuntu 16.04. Can I install Redmine 3.3 and stick with MySQL 5.7 now?

**#30 - 2017-05-18 22:56 - Deoren Moor**

Mark Anderson wrote:

| *Am about to make the move to Ubuntu 16.04. Can I install Redmine 3.3 and stick with MySQL 5.7 now?*

For what it is worth, we have been using an Ubuntu 16.04 + MariaDB 10.0.x setup for over six months now without any obvious issues.

- web server: Ubuntu 16.04, nginx/Passenger, mysql-client 5.7.x
- database server: MariaDB 10.0.x

**#31 - 2017-05-19 12:42 - Pavel Rosický**

- *File issue\_nested\_set.rb.patch added*

@Stephane Evr - Hi, the index on :issues => [:root\_id, :lft, :rgt] should be definitely unique.

To avoid duplicate entries during shifts I added an additional reorder statement:

```
remove_from_nested_set  
.reorder('lft desc')  
add_to_nested_set  
.reorder('lft asc')
```

but it didn't help anyway, I think it's because shifts are overlapping, especially during creating & deleting records at the same time.

unfortunately awesome\_nested\_set has the same issue

locking all issues instead of subtree works correctly (no deadlocks), but it should be definitely avoided for performance reasons  
self.class.reorder(:id).lock

log

# Running:

Deadlock detected on getting lock, restarting transaction retry #1 thread: 107872360  
Deadlock detected on update, restarting transaction retry #1 thread: 107872360  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863660  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107872360  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #2 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107872360  
Deadlock detected on update, restarting transaction retry #1 thread: 107872360  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on update, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863660  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107872360  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863880  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107863660  
Deadlock detected on getting lock, restarting transaction retry #1 thread: 107872360

..

Finished in 14.603111s, 0.1370 runs/s, 0.6848 assertions/s.

2 runs, 10 assertions, 0 failures, 0 errors, 0 skips

**#32 - 2018-08-08 10:39 - yossi edri**

Hi,  
what is the highest tested version of MySQL that is supported in Redmine 3.x - 4.x  
thanks

**#33 - 2018-08-19 16:38 - حـ**

yossi edri wrote:

Hi,  
what is the highest tested version of MySQL that is supported in Redmine 3.x - 4.x

| thanks

Salam

redmine 3.2 work by last version of mysql but redmine 3.3 and higher not work well with mysql 5.5 or higher

#### #34 - 2018-08-25 18:14 - Alexandr Kirilov

Just installed Redmine in following ports for FreeBSD. There are included mysql56, and the lowest version available from ports is mysql55. I've been trying Redmine with mysql80. Seems working for the case of mysql. But I got this issue - <http://www.redmine.org/boards/2/topics/55693>.

#### #35 - 2018-10-16 18:12 - Thomas Löber

Does it make sense to retry the transaction inside the Issue class?

In app/models/issue.rb:

```
def self.transaction(options={}, &block)
  retry_count = 0
  begin
    super
  rescue ActiveRecord::StatementInvalid => error
    raise if connection.adapter_name !~ /mysql/i
    raise if error.message !~ /Deadlock found when trying to get lock/
    raise if retry_count == 10
    retry_count += 1
    wait_ms = rand(retry_count * 100)
    if logger
      logger.info("Deadlock found when saving #{self}: "\
        "Waiting for #{wait_ms} ms before restarting the transaction (retry ##{retry_count})")
    end
    sleep(wait_ms / 1000.0)
  retry
  end
end
```

#### Files

mysql5.7_deadlocks_fix.patch	1.54 KB	2015-03-16	Jean-Philippe Lang
issue_nested_set.rb.patch	4.86 KB	2017-05-19	Pavel Rosický