I have a complex hierarchy of around 15000 issues in redmine, where an issue of this set could potentially have 3000 subtasks.

When doing CRUD operations on such issue, I noticed significant slow downs, caused by the `lock_nested_set` function.

Here is the profiling for the query actually run in `lock_nested_set`:

```
SELECT `issues`.id FROM `issues` WHERE (root_id IN (SELECT root_id FROM issues WHERE id IN (70395,70389)))
ORDER BY `issues`.id ASC FOR UPDATE;
```

```
+-------+
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>70371</td>
</tr>
<tr>
<td>70373</td>
</tr>
<tr>
<td>70375</td>
</tr>
<tr>
<td>70377</td>
</tr>
<tr>
<td>70379</td>
</tr>
<tr>
<td>70381</td>
</tr>
<tr>
<td>70383</td>
</tr>
<tr>
<td>70385</td>
</tr>
<tr>
<td>70387</td>
</tr>
<tr>
<td>70389</td>
</tr>
<tr>
<td>70391</td>
</tr>
<tr>
<td>70393</td>
</tr>
</tbody>
</table>
+-------+
2932 rows in set (2.70 sec)
```

```
mysql> show profile for QUERY 1;
+-------------------------------+------------+
<table>
<thead>
<tr>
<th>Status</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>starting</td>
<td>0.00025</td>
</tr>
<tr>
<td>Waiting for query cache lock</td>
<td>0.00004</td>
</tr>
<tr>
<td>checking query cache for query</td>
<td>0.000081</td>
</tr>
<tr>
<td>checking permissions</td>
<td>0.000003</td>
</tr>
<tr>
<td>checking permissions</td>
<td>0.000004</td>
</tr>
<tr>
<td>Opening tables</td>
<td>0.000038</td>
</tr>
<tr>
<td>System lock</td>
<td>0.000017</td>
</tr>
<tr>
<td>init</td>
<td>0.000056</td>
</tr>
<tr>
<td>optimizing</td>
<td>0.000013</td>
</tr>
<tr>
<td>statistics</td>
<td>0.000023</td>
</tr>
</tbody>
</table>
```
It takes around 3 seconds to execute the whole query.

I think the main problem is with the nested SELECT statement. If I execute it separately, then paste its results directly into the main query, the query is much faster:

```sql
mysql> SELECT root_id FROM issues WHERE id IN (70395,70389);
+---------+
| root_id |
+---------+
|   45083 |
+---------+
1 row in set (0.00 sec)

SELECT `issues`.`id` FROM `issues` WHERE (root_id IN (45083)) ORDER BY `issues`.`id` ASC FOR UPDATE;

+-------+
|       |
|       |
| 70371 |
| 70373 |
| 70375 |
| 70377 |
| 70379 |
| 70381 |
| 70383 |
| 70385 |
| 70387 |
| 70389 |
| 70391 |
| 70393 |
+-------+
2932 rows in set (0.01 sec)
```
I am not an expert in sql queries, and don't want to break anything... Shouldn't we use a JOIN instead?

Environment:
- Redmine version: 3.3.0.stable
- Ruby version: 2.2.2-p95 (2015-04-13) [x86_64-linux]
- Rails version: 4.2.6
- Environment: development
- Database adapter: Mysql2

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

Associated revisions

Revision 15891 - 2016-10-08 11:15 - Jean-Philippe Lang

#lock_nested_set very slow on mysql with thousands of subtasks (#23318).

Patch by Stephane Evr.

Revision 15892 - 2016-10-09 10:37 - Jean-Philippe Lang

Reverts r15891 (#23318).

Deadlocks with MySQL.

Revision 16053 - 2016-12-10 09:54 - Jean-Philippe Lang

#lock_nested_set very slow on mysql with thousands of subtasks (#23318).

Patch by Stephane Evr.

Revision 16054 - 2016-12-10 10:44 - Jean-Philippe Lang

Reverted r16053 (#23318).

SQL error with PostgreSQL.

History

#1 - 2016-07-13 19:22 - Stephane Evr

FYI, I was able to change the statement used in lock_nested_set

from:

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


to:

```
2020-04-17
```
so far the later returns instantly, with the same results as the former. I don’t know though how it affects the locking mechanism.

#2 - 2016-07-14 05:34 - Toshi MARUYAMA
- Category set to Database

#3 - 2016-07-16 10:10 - Go MAEDA
- Target version set to Candidate for next major release

Passed all tests.

The following is a diff of changes made by Stephane Evr.

```
Index: lib/redmine/nested_set/issue_nested_set.rb
===================================================================
--- lib/redmine/nested_set/issue_nested_set.rb (revision 15663)
+++ lib/redmine/nested_set/issue_nested_set.rb (working copy)
@@ -158,7 +158,7 @@
    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
+    self.class.reorder(:id).joins("INNER JOIN #{self.class.table_name} t2 ON #{self.class.table_name}.root_id = t2.root_id").where("t2.id IN (?)", sets_to_lock).distinct.lock.ids
 end
 end
```

#4 - 2016-07-16 10:18 - Toshi MARUYAMA
- Target version changed from Candidate for next major release to 3.4.0

#5 - 2016-10-08 11:07 - Jean-Philippe Lang
- Subject changed from lock_nested_set very slow on mysql to #lock_nested_set very slow on mysql with thousands of subtasks

Go MAEDA wrote:

```
[Passed all tests.]
```

Did you run the tests with mysql? Because I get an error with Postgresql with the patch applied (FOR UPDATE not allowed with DISTINCT). Tests pass without .distinct

#6 - 2016-10-08 11:10 - Go MAEDA
Jean-Philippe Lang wrote:

Did you run the tests with mysql? Because I get an error with Postgresql with the patch applied (FOR UPDATE not allowed with DISTINCT).
Tests pass without .distinct

Sorry, I run the tests only with sqlite.

---

**#7 - 2016-10-08 11:16 - Jean-Philippe Lang**
- Category changed from Database to Performance
- Status changed from New to Closed
- Assignee set to Jean-Philippe Lang
- Resolution set to Fixed

Patch committed without the .distinct call, thanks.

---

**#8 - 2016-10-09 06:48 - Toshi MARUYAMA**
- Status changed from Closed to Reopened

MySQL "IssueNestedSetConcurrencyTest#test_concurrency" and "IssueNestedSetConcurrencyTest#test_concurrent_subtasks_creation" fail.
http://www.redmine.org/builds/logs/build_trunk_mysql_ruby-2.3_3063.html

It may be related with #19344.

---

**#9 - 2016-10-09 06:49 - Toshi MARUYAMA**
- Related to Defect #19344: MySQL 5.6: IssueNestedSetConcurrencyTest#test_concurrency : always fails added

---

**#10 - 2016-10-09 10:38 - Jean-Philippe Lang**
- Assignee deleted (Jean-Philippe Lang)
- Target version deleted (3.4.0)

r15891 reverted.

---

**#11 - 2016-10-12 19:49 - Stephane Evr**

I found an alternative solution which passes the tests, although not as fast as with a JOIN. Here, I simply group by issue id in the original subquery:

```
SELECT `issues`.* FROM `issues` WHERE (root_id IN (SELECT root_id from `issues` WHERE id IN (96457,96455) GROUP BY id) ORDER BY `issues`.
```

So the statement would be rewritten as:

```ruby
self.class.reorder(id).where("root_id IN (SELECT root_id FROM #{self.class.table_name} WHERE id IN (?) GROUP BY id)",
sets_to_lock).lock.ids
```
I finally found how to refactor the JOIN Statement and keep the initial performance improvements. Please find attached a patch against the master branch. This passed the issue_nested_set_concurrency_test on MySQL.

I ran the test 10 times and all passed.

Reverted r16053 because of SQL error with PostgreSQL.

PostgreSQL does not accept the DISTINCT in the subquery:
http://www.redmine.org/builds/logs/build_trunk_postgresql_ruby-2.3_3124.html

One option would be to have a statement specific to MySQL just like we already have one for SQL Server.

Perhaps this is the solution. I don't know if PostgreSQL is affected by this problem.

Based #note-13 patch.
This patch reduces #19344 test failure times on my CentOS7 mariadb-5.5.52-1.el7.x86_64.
(Not always test pass. About 50% time test passes.)
<table>
<thead>
<tr>
<th>Files</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>issue_nested_set.patch</td>
<td>834 Bytes</td>
<td>2016-10-13</td>
<td>Stephane Evr</td>
</tr>
<tr>
<td>mysql-deadlock-02.diff</td>
<td>1.31 KB</td>
<td>2017-03-21</td>
<td>Toshi MARUYAMA</td>
</tr>
</tbody>
</table>