Redmine - Defect #15870

Parent task completion is 104% after update of subtasks

2014-01-15 02:31 - joe chen

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Jean-Philippe Lang	**% Done:**	0%
Category:	Issues	**Estimated time:**	0.00 hour
Target version:	2.4.3		
Resolution:	Fixed	**Affected version:**	2.3.3

Description

What is the problem caused by?
How to solve this problem?
How to avoid this problem again?
Hope to receive your reply as soon as possible.
Thank you!

Related issues:

Has duplicate Redmine - Defect #15974: percentage of done is more than 100%.	**Closed**

Associated revisions

Revision 12669 - 2014-01-19 09:35 - Jean-Philippe Lang

Make sure that we don't set the done ratio to a value > 100% (#15870).

Revision 12739 - 2014-01-30 23:18 - Jean-Philippe Lang

Reverts r12669 and add a test for #15870.

Revision 12745 - 2014-01-31 18:42 - Jean-Philippe Lang

Fixed calculation of done_ratio of parent task with child having estimated_time at 0 (#15870).

Revision 12847 - 2014-02-06 19:00 - Jean-Philippe Lang

Merged r12745 (#15870).

History

#1 - 2014-01-15 14:24 - Etienne Massip

- Category set to Issues

- Priority changed from Urgent to Normal

Could you please give more details according to SubmittingBugs?

#2 - 2014-01-16 08:29 - Toshi MARUYAMA

- Status changed from New to Needs feedback

#3 - 2014-01-17 08:38 - joe chen

Etienne Massip wrote:

> Could you please give more details according to SubmittingBugs?

It is a parent task.
The parent task has a huge number of subtasks.
The completion of tasks of the parent is automatically calculated as 104%.

#4 - 2014-01-19 09:37 - Jean-Philippe Lang

- Subject changed from Task completion is 104% to Parent task completion is 104% after update of subtasks

- Status changed from Needs feedback to Resolved

- Assignee set to Jean-Philippe Lang

- Target version set to 2.4.3

- Resolution set to Fixed

Fixed in r12669.

#5 - 2014-01-27 15:57 - Daniel Felix

- Has duplicate Defect #15974: percentage of done is more than 100%. added

#6 - 2014-01-27 16:20 - Bruno Medeiros

r12669 doesn't seem a real fix to me, it only forces the value to 100 if it's bigger than 100, it doesn't fix the calculation problem that lead to the incorrect value in the first place. Am I missing something?

#7 - 2014-01-30 16:13 - Uwe Koloska

No, you are absolutely right. The wrong calculation is not caused by some rounding errors or something like this, it comes from subtasks without estimated time while others have.

Scenario:
Parent
- Sub1: 40h
- Sub2: 40h
- Sub3: 20h
- Sub4: --

when closing all subtickets, done sums up to 125.

But now for the tricky part: To solve this, you can set Sub4 to 0.01. Then done is correctly calculated with 100.

And now for the fun fact: if you now change Sub4 back to nothing in estimated time, the calculation is correct:
- Sub1: 40h := 30%
- Sub2: 40h := 30%
- Sub3: 20h := 15%
- Sub4: --- := 25%

Ahh -- but if you change Sub4 to 0h you get the wrong result 125% again. Maybe the estimated time is not really gone if you delete just the entry?

possible solutions

- if there is at least one subticket with an estimated time, then treat all unset times as 0h and don't include this tickets %done in the calculation for the parent ticket
- calculate the unestimated tickets with their weight coming from 1/number_of_subtickets and give the rest to the estimated tickets

The last one is my preferred solution.

#8 - 2014-01-30 23:32 - Jean-Philippe Lang

Uwe Koloska wrote:

> Scenario:
> Parent
> - Sub1: 40h
> - Sub2: 40h
> - Sub3: 20h
> - Sub4: --
>
> when closing all subtickets, done sums up to 125.

I've added a test for this scenario and reverted r12669, but it passes (see r12739). Any ideas?

#9 - 2014-01-31 08:40 - Daniel Felix

As I already said some time ago, the database column estimated_hours has the datatype real, which is one of the worst datatypes.

This should be changed to a much better datatype like decimal. Well in some edgecases float would be good enough to. But we never use real in any software/database application in our whole company as it is just awful. Yes real is a bit faster, and a bit smaller in some cases than a high precision decimal value. But well, disk space is cheaper than 1990 and a CPU easily could handle this.

Maybe you would give it a try and change it to a decimal value like decimal(13,5) which gives you 8 predecimal and 5 decimal digits. This would be enough for the most cases.

#10 - 2014-01-31 08:55 - Jan Niggemann (redmine.org team member)

Daniel Felix wrote:

> Well in some edgecases float would be good enough

FLOAT is an approximate numeric data type whereas DECIMAL isn't, at least in MySQL and MSSQL.

> Maybe you would give it a try and change it to a decimal value like decimal(13,5) which gives you 8 predecimal and 5 decimal digits.

Good idea

#11 - 2014-01-31 11:17 - Uwe Koloska

Jean-Philippe Lang wrote:

> I've added a test for this scenario and reverted r12669, but it passes (see r12739). Any ideas?

Yes:

```
      Issue.generate!(:estimated_hours => 40, :parent_issue_id => parent.id)
      Issue.generate!(:estimated_hours => 40, :parent_issue_id => parent.id)
      Issue.generate!(:estimated_hours => 20, :parent_issue_id => parent.id)
-     Issue.generate!(:parent_issue_id => parent.id)
+     Issue.generate!(:estimated_hours => 0,  :parent_issue_id => parent.id)
```

At least this is the case, where I can reproduce the error.

But there is something in the story that makes me think (*error* means %done > 100% and all changes done to issue4):

1. error in normal operation if issue4 has no estimated time (the field was not touched and looks empty)
2. no error if estimated time is set to 0.01
3. **nor error** if estimated time is deleted (and stay visually empty when reediting)
4. error if estimated time is set to 0
5. no error if estimated time is deleted again

#12 - 2014-01-31 13:37 - Jean-Philippe Lang

It's a bit different from the above and it fails. The following change should fix it:

```
Index: app/models/issue.rb
===================================================================
--- app/models/issue.rb (révision 12743)
+++ app/models/issue.rb (copie de travail)
@@ -1355,7 +1355,7 @@
       unless Issue.use_status_for_done_ratio? && p.status && p.status.default_done_ratio
         leaves_count = p.leaves.count
         if leaves_count > 0
-          average = p.leaves.average(:estimated_hours).to_f
+          average = p.leaves.where("estimated_hours > 0").average(:estimated_hours).to_f
           if average == 0
             average = 1
           end
```

Could you give it a try before I commit?

#13 - 2014-01-31 13:55 - Daniel Felix

I retried the described problems. Here is the solution:

In this testcase the error occurs with the given scenario of 4 tickets (40,40,20,NULL).
Here is the MS SQL Server Statement to reproduce. Shouldn't be a problem to change this for different dbms.

Issue 57 is the last issue with Null or 0 Value.

```
  -- first scenario which provides the error.
  update redmine_development.dbo.issues
  set estimated_hours = 0
  WHERE ID = 57

  -- second scenario which provides the error.
  update redmine_development.dbo.issues
```

```
  set estimated_hours = NULL
  WHERE ID = 57

  -- Code to retry the Rails calculation
  declare @count int

  select @count = count(*)
  -- select *
  FROM [redmine_development].[dbo].[issues]
  WHERE parent_id = 53

  declare @avg decimal(13,5)

  select @avg = avg(estimated_hours)
  -- select *
  FROM [redmine_development].[dbo].[issues]
  WHERE parent_id = 53

  declare @val decimal(13,5)

  SELECT @val = sum(d.v) FROM (
    -- For testing purposes as subquery
    SELECT COALESCE(CASE WHEN estimated_hours > 0 THEN estimated_hours ELSE NULL END, @avg)
    * (CASE WHEN is_closed = 1 THEN 100 ELSE COALESCE(done_ratio, 0) END) as v
    FROM [redmine_development].[dbo].[issues] as i
    inner join [redmine_development].[dbo].[issue_statuses] as ist
        on i.status_id = ist.id
    WHERE parent_id = 53
  ) as d

  set @val = @val / (@avg * @count)
  select @val
```

The error is quite easy.
If estimated_hours is NULL, the AVG-aggregation will leave the data row out in its calculation, which means:
4 data rows including one data row with NULL value results in this calculation: (40 + 40 + 20) / 3, means that this calculation (source:trunk/app/models/issue.rb#L1358) resulting in 33,333... %
The sum of all estimated hours results in 10000 in both cases, test case 1 and test case 2.

This means later that this calculation (source:trunk/app/models/issue.rb#L1365) results in this result:

```
progress = done / (average * leaves_count)
progress = 10000 / (33.333 * 4)
progress = 75.00001
```

This would be the correct value.

The same thing with a 0 in estimated_hours, causes avg to take this value into account which results in this calculation (40 + 40 + 20 + 0) / 4 which means an avg of 25.

Taking this into account to the previous calculation results in this:

```
progress = done / (average * leaves_count)
progress = 10000 / (25 * 4)
progress = 125.00000
```

BAM! Mistery resolved. :-P

This leading me to this solution...
Replacing this line (source:trunk/app/models/issue.rb#L1358) with this, should solve the problem:

```
p.leaves.where("estimated_hours <> NULL").average(:estimated_hours).to_f
```

or better with a quoted_nil:

```
p.leaves.where("estimated_hours <> #{connection.quoted_nil}".average(:estimated_hours).to_f
```

Can you please try this, this should solve the problem.

Best regards,
Daniel

#14 - 2014-01-31 13:56 - Daniel Felix

Damn... Jean-Philippe has posted earlier. :-D

#15 - 2014-01-31 15:27 - Uwe Koloska

Jean-Philippe Lang wrote:

> It's a bit different from the above and it fails. The following change should fix it:
>
> [...]
>
> Could you give it a try before I commit?

I have changed my test installation and the fix gives the correct result for the aforementioned use case.

Thank you!

#16 - 2014-01-31 17:27 - Holger Just

Note that this issue (and the proposed solutions) are the same as #15957 I posted a week ago with a patch and a test.

Unfortunately, I don't have the right to add issue relations here. My approach there reused part of the SQL already used in that function. In the end, the solution is to exclude all 0 values from the average calculation and to treat them as NULL.

#17 - 2014-01-31 18:51 - Jean-Philippe Lang

Fix committed in r12745.

Daniel, thanks for the explanation. Children with estimated time at 0 were taken into account for the average calculation but treated as null values after that. Holger, sorry for not having reviewed your patch earlier. r12745 should give the very same results with a slightly cleaner query.

#18 - 2014-02-06 19:00 - Jean-Philippe Lang

- Status changed from Resolved to Closed

Files

q1.jpg	10.6 KB	2014-01-15	joe chen