Refactoring Systems with Confidence

Jesse Toth and Nathan Witmer
OSCON 2015
Domain Model
Permissions
access control
search result filtering
event feeds
repository.pullable_by?(user)
Organic growth and complexity
Performance problems
SELECT r.id FROM r,  
SELECT r.id as r_ids, 2 as perms from r  
WHERE r.owner_id = 99999  
AND (r.x = 0)  
UNION ALL  
SELECT r.id as r_ids, 1 as perms from r  
INNER JOIN p ON r.id = p.r_id  
WHERE p.u_id = 99999  
AND (r.x = 0)  
UNION ALL  
SELECT r.id as r_ids, 2 as perms from r  
INNER JOIN t ON r.o_id = t.o_id  
INNER JOIN t_m ON t_id = t_m.t_id  
WHERE t.name = 'X'  
AND t.m.u_id = 99999  
AND (r.x = 0)  
UNION ALL  
SELECT r.id as r_ids, GROUP_CONCAT(distinct t.p) as perms from r  
INNER JOIN t_m r_t ON r.id = r_t.r_id  
INNER JOIN t ON r_t.t_id = t.id  
INNER JOIN t_m t_m_u_t ON t.id = u_t.t_id  
WHERE u_t.u_id = 99999  
AND t.name != 'X'  
AND t.p in (2, 1, 0)  
AND (r.x = 0)  
GROUP BY r.id  
UNION ALL  
SELECT r.id as r_ids, 0 as perms from r  
JOIN u ON r.plan_owner_id = u.id  
JOIN t ON t.o_id = u.id  
JOIN t_m ON t.id = t_m.t_id  
WHERE u.type = 'XX'  
AND t.name = 'X'  
AND t.m.u_id = 99999  
AND (r.x = 0)  
AND r.parent_id IS NOT NULL AS unioned  
WHERE r.id = r_ids;
Edge cases and transitional states
Difficult to build on

Organizations Next

One of the GitHub goals we've been talking about lately is, "It's easier to work together than alone". As a long term goal for Organizations, I think this is perfect: people should love Organizations because it makes it easy for them to build cool shit with their coworkers.
A New Permissions System
Goals
Simple, flexible interface
Fast lookups
Easy to integrate and operate
Abilities
Actor → action → Subject
User → read → Team

Actor → action → Subject

User → read → Team
SELECT 1 FROM abilities
WHERE actor_id = user_id
    AND actor_type = 'User'
    AND subject_id = repository_id
    AND subject_type = 'Repository'

user.can? :read, repository
SELECT subject_id
FROM abilities
WHERE actor_id = user_id
AND actor_type = 'User'
AND subject_type = 'Repository'
Replacing the permissions system
Refactoring with Scientist
You can’t be confident that test cases fully cover the complexity of real-world data
If test coverage is thin, you can’t be sure that the tests you add fully cover all behavior — especially if the intended behavior is not 100% clear in the collective knowledge of the team.
Current behavior may have unintended bugs and side-effects that users are relying on!
Test suites don’t cover production performance
Production data is the real test
Compare return values

Legacy Code

Refactored Code

repository.pullable_by?(user)

execute

return result

publish

Compare return values

Metrics
Scientist

A Ruby library for carefully refactoring critical paths
class Repository

    def pullable_by?(user)
        # old code...
        end

    end

end
class Repository

  def pullable_by?(user)
    pullable_by_legacy?(user)
  end

  def pullable_by_legacy?(user)
    # old code
  end

end
class Repository

  def pullable_by?(user)
    pullable_by_legacy?(user)
  end

  def pullable_by_legacy?(user)
    # old code
  end

  def pullable_by_refactored?(user)
    # new code
  end

end
class Repository
  include Scientist

  def pullable_by?(user)
    pullable_by_legacysyzygth((user))
  end

  def pullable_by_legacy?(user)
    # old code
  end

  def pullable_by_refactored?(user)
    # new code
  end
end
class Repository
  include Scientist

  def pullable_by?(user)
    # Let's do an experiment
    science "repository.pullable_by" do |experiment|
      end
    end
  end
end
class Repository
  include Scientist

  def pullable_by?(user)
    # Let's do an experiment
    science "repository.pullable_by" do |experiment|

      # Return this value no matter what
      experiment.use { pullable_by_legacy?(user) }

    end
  end
end
class Repository
  include Scientist

  def pullable_by?(user)
    # Let's do an experiment
    science "repository.pullable_by" do |experiment|

      # Return this value no matter what
      experiment.use { pullable_by_legacy?(user) }

      # Run the new code too, and compare the results
      experiment.try { pullable_by_refactored?(user) }

    end
  end
end
class Repository
  include Scientist

  def pullable_by?(user)
    # Let's do an experiment
    science "repository.pullable_by" do |experiment|
      # Return this value no matter what
      experiment.use { pullable_by_legacy?(user) }

      # Run the new code too, and compare the results
      experiment.try { pullable_by_refactored?(user) }

      # Some context for published results
      experiment.context :user => user, :repo => self
    end
  end
end
class Repository
  include Scientist

  def pullable_by?(user)
    # Let's do an experiment
    science "repository.pullable_by" do |experiment|
      # Return this value no matter what
      experiment.use { pullable_by_legacy?(user) }
      # Run the new code too, and compare the results
      experiment.try { pullable_by_refactored?(user) }
      # Some context for published results
      experiment.context :user => user, :repo => self
    end
  end
end
- num times the experiment has run
- num times the use and try blocks’ return values differed
- timings of each block’s execution

- return values of use and try blocks for experiments that mismatched
### Active science experiments

These experiments are running now. Don't forget that experiments can hurt site performance!

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Rate</th>
<th>Status</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>ability.organization.adminable_by</td>
<td>2,060/min</td>
<td>1%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.organization.all-team-members</td>
<td>463/min</td>
<td>10%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.organization.can-create-repository</td>
<td>10/min</td>
<td>10%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.repository.adminable-by</td>
<td>1,319/min</td>
<td>1%</td>
<td>&lt;1% wrong</td>
</tr>
<tr>
<td>ability.repository.members</td>
<td>7,048/min</td>
<td>10%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.repository.member_ids</td>
<td>506/min</td>
<td>100%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.repository.pullable-by</td>
<td>2,420/min</td>
<td>1%</td>
<td>&lt;1% wrong</td>
</tr>
<tr>
<td>ability.repository.pushable-by</td>
<td>1,936/min</td>
<td>1%</td>
<td>&lt;1% wrong</td>
</tr>
<tr>
<td>ability.team.members</td>
<td>999/min</td>
<td>1%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.team.repositories</td>
<td>20/min</td>
<td>1%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.team.user-ids-for</td>
<td>427/min</td>
<td>25%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.user.organization-filter</td>
<td>99/min</td>
<td>1%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.user.staff</td>
<td>8/min</td>
<td>100%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>ability.user.teams</td>
<td>27,311/min</td>
<td>25%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>api.has-access</td>
<td>15/min</td>
<td>100%</td>
<td>0% wrong</td>
</tr>
<tr>
<td>issues.elasticsearch-label-count</td>
<td>453/min</td>
<td>100%</td>
<td>7% wrong</td>
</tr>
<tr>
<td>plan_owner.repository.plan_owner</td>
<td>488/min</td>
<td>1%</td>
<td>&lt;1% wrong</td>
</tr>
<tr>
<td>Participation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0% 1% 10% 25% 50% 75% 100%</td>
<td>of calls will run the experiment.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Accuracy

The number of times that the candidate and the control agree or disagree. View mismatches
The number of incorrect/ignored only.
Performance

The 75th/99th percentile durations of the control and the candidate, in milliseconds.
Control is [8109865, 8709932, 8738177, 12892765, 17645968]

Candidate is missing [8709932]

Show raw data

Control is [4106194, 4106426, 4797677, 4810634, 5048914, 5127427, 6003747, 6907747, 7389532, 7389500, 7635630, 7970132, 8303666, 8736200, 9903600, 10634086, 10776294, 10887123, 12062945, 12632760, 13152695, 13401456, 13485873, 14361115, 14696212, 15200535, 15214500, 15303998, 15416079, 15525715, 16111906, 16329693, 16540083, 17009632, 17011391, 17045173, 17433149, 17523899, 18352073, 18478587, 19388105, 19399069, 19540405, 1958076, 19638293, 19799279, 2079807, 20913308, 20710460, 20786305, 20808698, 2108203, 21146724, 21183112, 21630923, 21632234, 21887808, 21932718, 22396416, 22440260, 22699542, 22737575, 22775112, 22849681, 22927761, 22932432, 22932728, 23231782, 23413631, 23605961, 23721897, 23852602, 23937684, 24181656, 24131832, 24344593, 24348275, 24618795, 24663991, 24914235, 24962616, 24967070, 25021875, 25222043, 25231723, 25272603, 25282047, 25370221, 25495114, 25495232, 25665502, 25940821, 25988784, 25992412, 26027571, 26651658, 26983322, 27154735, 27191534, 27567264, 27733819, 27793521, 27793548... (truncated)

Candidate is missing [5127427]

Show raw data
Backfill and Validation
Legacy Permissions

 Abilities

write

read & verify
Legacy Permissions

Abilities

write
read & verify
backfill
Legacy Permissions:
- write
- read & verify
- backfill
- repair

Abilities:
Legacy Permissions

Abilities

- write
- read & verify
- backfill
- repair
Legacy Permissions

Abilities

write
read & verify
backfill & validation
repair
Data quality: remove repos from teams belonging to orgs not associated with the repo
#26145 opened on May 17 by jesseplusplus

Private repository transfers to organizations preserve collaborators on forks
#18343 opened on Dec 10, 2013 by zerowidth

Data quality: org collaborators on repositories
#22500 opened on Mar 5 by zerowidth

Data Quality: Fix for invalid Team permissions
#15816 opened on Oct 9, 2013 by zerowidth

data quality: orphaned teams bug doubledown
#26124 opened on May 16 by jesseplusplus

Abilities mismatch: leftover teams from previous org after transfer
#25550 opened on May 5 by jesseplusplus

Repo network data quality: private fork / plan owner mismatch
#24355 opened on Apr 11 by zerowidth

Abilities mismatch: organization_id data quality issue
#25345 opened on Apr 30 by jesseplusplus

Repo network data quality: private root / plan owner mismatch
#24319 opened on Apr 10 by zerowidth

Repository network data quality issues
#23998 opened on Apr 3 by jesseplusplus

Ensure plan owner and parent are updated correctly when reparenting forks
#23587 opened on Mar 26 by jesseplusplus

Data quality: Repository transfers leave orphaned watchers
#22353 opened on Mar 3 by zerowidth
Legacy Permissions

- write
- read & verify
- backfill & validation
- repair

Abilities
The Results
Improved organization permissions

June 10, 2015  connors  New Features  Edit

Organizations have always been the best way for teams to work together and collaborate on code. We’re happy to announce major improvements to GitHub organization permissions. These improvements include new customizable member privileges, fine-grained team permissions, and more open communication.

Powerful collaboration with improved organization permissions
Lessons Learned
Production Data *is the* Real Test
Data Quality is Paramount
Math is Important
1 Team with 10,000 Users × 5,000 Repositories
1 Team with 10,000 Users × 5,000 Repositories = 50,000,000 rows
YO DAWG,
I HEARD YOU LIKE SCIENCE...
Double-Check your Queries
Team repository permissions outage (2014-07-28) #118

Closed

tmml opened this issue on Jul 29, 2014 • 28 comments

14:15 UTC

tmml commented on Jul 29, 2014

The migration in github/github#29673 accidentally deleted all rows in the team_members join table with repository_id IS NOT NULL. Repositories added to teams directly are currently inaccessible to users on those teams.

8:50 UTC

Everything operating normally.

8:00 UTC

We have identified the root cause and are now restoring access to customer repositories.

7:32 UTC

We are investigating an issue preventing repository access for some customers.
More Uses of Scientist
Pipeline to render HTML, Markdown, and Blobs
Load-testing a new search cluster
Testing performance and correctness of query changes
Thanks! ❤️