Redmine - Feature #38402

"Any searchable text" filter for issues

2023-04-02 10:10 - Go MAEDA

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Go MAEDA	**% Done:**	0%
Category:	Issues filter	**Estimated time:**	0.00 hour
Target version:	5.1.0		
Resolution:	Fixed		

Description

The attached patches add a new issues filter "Any searchable text". Unlike existing filters that check a specific single field, this new filter checks all text-type core fields (subject, description, notes) and searchable custom fields. For example, if you apply the filter "[Any searchable text] [contains] [recipe]", the issues list shows issues that contain the word "recipe" in the subject, description, notes, or searchable custom fields. This is almost the same behavior as the search box in the upper right corner of the page. Actually, this new filter uses Redmine::Search::Fetcher, the code of the existing search box.

The new "Any searchable text" is useful when you want to find issues that contain a specific keyword but you don't know which field contains the keyword. In Redmine 5.0, you have to try one filter after another, such as "Subject," "Description," "Notes", and custom fields. With the new filter, you can find issues that contain the keyword in any of the fields in a single operation.

What is different from the search box in the upper right corner of the page is that this is a filter. This means that you can further refine the result by applying other filters such as the status, assignee, and target version.

clipboard-202304021708-v4fbz.png

Related issues:

Related to Redmine - Feature #3491: Filtering while searching for issues	**Closed**	**2009-06-13**
Related to Redmine - Feature #10574: Add filtering ability to search or searc...	**Closed**	
Related to Redmine - Feature #14943: Split the 'Status' search field	**Closed**	
Related to Redmine - Feature #9180: Improve search system for issues - like "...	**Closed**	**2011-09-04**
Has duplicate Redmine - Feature #680: free text ticket filter	**Closed**	**2008-02-17**

Associated revisions

Revision 22164 - 2023-04-05 11:22 - Go MAEDA

Add "Any searchable text" filter for issues (#38402).

Patch by Go MAEDA and Holger Just.

Revision 22165 - 2023-04-05 11:23 - Go MAEDA

Group text filters in the filter select list (#38402).

Patch by Go MAEDA.

Revision 22166 - 2023-04-05 11:24 - Go MAEDA

Reorder groups in the filter select list (#38402).

Patch by Go MAEDA.

Revision 22167 - 2023-04-05 11:26 - Go MAEDA

Update locales (#38402).

Revision 22168 - 2023-04-05 15:52 - Go MAEDA

Fix a bug the "doesn't contain" operator returns no issues if the search finds no issues (#38402).

Patch by Holger Just.

Revision 22169 - 2023-04-06 01:36 - Go MAEDA

Change the behavior of the "Any searchable text" filter from OR search to AND search (#38402).

Patch by Go MAEDA.

Revision 22172 - 2023-04-06 12:08 - Go MAEDA

Fixed an issue where an empty group could appear in the filter selection list caused by r22166 (#38402).

Revision 22203 - 2023-04-18 14:05 - Go MAEDA

Fix "Any searchable text" filter doesn't support the project filter with the value "my projects" or "my bookmarks" (#38402).

Revision 22221 - 2023-05-01 16:33 - Go MAEDA

Make the "Any searchable text" filter search only open issues when the status filter is set to "open" (#38402).

History

#1 - 2023-04-02 16:30 - Go MAEDA

- Related to Feature #3491: Filtering while searching for issues added

#2 - 2023-04-03 23:33 - Holger Just

I would love to have this feature incorporated into Redmine. I think it would be very useful!

As a slight addition though: As a search over many objects can be quite expensive, it might be worthwhile to further filter the projects passed to Redmine::Search::Fetcher.new, similar to the rules in Query#project_statement, so that the search results are only fetched from the selected and visible (sub-)projects of the query rather than all (sub-)projects. While this does not change anything functionally (as the actual returned issue query result is further restricted by the actual project filter), it may significantly improve the performance of filtered queries in large Redmines. This would also avoid searching in archived projects or projects not visible to the current user, whose issues we would never return.

This could be implemented as a new method on the IssueQuery model which could look like this:

```
def projects
  if project
    subprojects = project.descendants.allowed_to(:view_issues)
    if has_filter?("subproject_id")
      case operator_for("subproject_id")
      when '='
        # include the selected subprojects
        [project] + subprojects.where(id: values_for("subproject_id").map(&:to_i))
      when '!'
        # exclude the selected subprojects
        [project] + subprojects.where.not(id: values_for("subproject_id").map(&:to_i))
      when '!*'
        # main project only
        [project]
      else
        # all subprojects
        project_ids = [project] + subprojects
      end
    elsif Setting.display_subprojects_issues?
      [project] + subprojects
    else
      [project]
    end
  else
    projects = Project.allowed_to(:view_issues)
    case operator_for("project_id")
    when '='
      # include the selected subprojects
      projects.where(id: values_for("project_id").map(&:to_i)).to_a
    when '!'
      # exclude the selected subprojects
      projects.where.not(id: values_for("subproject_id").map(&:to_i)).to_a
    else
      # All projects
      nil
    end
  end
end
```

(Note that I have not fully tested this yet)

What do you think?

#3 - 2023-04-04 08:10 - Go MAEDA

Holger Just wrote in #note-2:

> As a slight addition though: As a search over many objects can be quite expensive, it might be worthwhile to further filter the projects passed to Redmine::Search::Fetcher.new, similar to the rules in Query#project_statement, so that the search results are only fetched from the selected and visible (sub-)projects of the query rather than all (sub-)projects.

Thank you for reviewing the patches and suggesting an improvement! The improvement makes the code much more efficient.

However, Redmine will have very similar codes in two places after adding the suggested IssueQuery#projects. It would be better if we can somehow use Query#project_statement.

I think that changing 0001-Add-Any-serchable-text-filter-for-issues.patch as follows can avoid the code dupulication and reduce the number of projects to search. What is your opinion?

```
diff --git a/app/models/issue_query.rb b/app/models/issue_query.rb
index 4db44b6b3..11f3f2adb 100644
--- a/app/models/issue_query.rb
+++ b/app/models/issue_query.rb
@@ -778,7 +778,11 @@ class IssueQuery < Query

   def sql_for_any_searchable_field(field, operator, value)
     question = value.first
-    projects = project&.self_and_descendants
+    if project || operator_for('subproject_id').present?
+      projects = Project.where(project_statement).allowed_to(:view_issues)
+    else
+      projects = nil
+    end
     fetcher = Redmine::Search::Fetcher.new(
       question, User.current, ['issue'], projects, attachments: '0'
     )
```

#4 - 2023-04-04 12:38 - Holger Just

Turns out, there were a couple of bugs / typos / copy-and-paste errors in my proposed method. Sorry for that :/ In any case, your proposal would allow for almost the same effect with much less new code so it is indeed much better!

Your proposal however does not handle global issue queries unfortunately. Here, we don't have a subproject_id filter but a separate project_id filter (which is added to the query statement with a generic sql_for_field call). To restrict the searched projects on global issue queries, we should also handle that case.

In my updated proposal below, we use the project_statement in case we are a project-local query as you have proposed. If I have understood the logic in Query#project_statement correctly, it will always return an SQL snippet if there is a project, potentially further restricted according to the subproject_id filter. Thus, I don't think we have to check for the presence of the subproject_id filter in sql_for_any_searchable_field.

For the global case, where we don't have a project for the query, we can restrict projects according to the project_id filter with the sql_for_field method. The project_id filter is defined to have only the = and ! operators (although I don't think this is verified during query generation), so this should generate a simple statement we can use in the global case.

```
  def sql_for_any_searchable_field(field, operator, value)
    question = value.first

    project_scope = Project.allowed_to(:view_issues)
    if project
      projects = project_scope.where(project_statement)
    elsif has_filter?('project_id')
      projects = project_scope.where(sql_for_field('project_id', operator_for('project_id'), values_for(
'project_id'), Project.table_name, 'id'))
    else
      projects = nil
    end

    fetcher = Redmine::Search::Fetcher.new(
      question, User.current, ['issue'], projects, attachments: '0'
    )
    # ...
```

What do you think?

#5 - 2023-04-04 15:37 - Go MAEDA

- Target version set to 5.1.0

Thank you. Setting the target version to 5.1.0!

#6 - 2023-04-05 11:27 - Go MAEDA

- Status changed from New to Closed

- Assignee set to Go MAEDA

- Resolution set to Fixed

Committed the patches.

#7 - 2023-04-05 13:33 - Holger Just

- Status changed from Closed to Reopened

- Resolution deleted (Fixed)

I was just having another look at r22164. Are you sure that the logic is correct for the !~ filter in case we find no issues with the search?

If I'm reading the code correctly, we now restrict the final query result to also return no issues (by returning '1=0') in that case. However, I think we should probably not restrict the result at all in that case?

As such, shouldn't the final part of the IssueQuery#sql_for_any_searchable_field method look like this with accompanying tests?

```
diff --git a/app/models/issue_query.rb b/app/models/issue_query.rb
index 1c60c6b61f..e455481215 100644
--- a/app/models/issue_query.rb
+++ b/app/models/issue_query.rb
@@ -799,7 +799,7 @@ def sql_for_any_searchable_field(field, operator, value)
       sw = operator == '!~' ? 'NOT' : ''
       "#{Issue.table_name}.id #{sw} IN (#{ids.join(',')})"
     else
-      '1=0'
+      operator == '!~' ? '1=1' : '1=0'
     end
   end

diff --git a/test/unit/query_test.rb b/test/unit/query_test.rb
index 2b02a1e07d..c5b407b99f 100644
--- a/test/unit/query_test.rb
+++ b/test/unit/query_test.rb
@@ -844,7 +844,7 @@ def test_filter_notes_should_ignore_private_notes_that_are_not_visible
     assert_equal [1, 3], find_issues_with_query(query).map(&:id).sort
   end

-  def test_fileter_any_searchable
+  def test_filter_any_searchable
     User.current = User.find(1)
     query = IssueQuery.new(
       :name => '_',
@@ -859,7 +859,52 @@ def test_fileter_any_searchable
     assert_equal [1, 2, 3], result.map(&:id).sort
   end

-  def test_fileter_any_searchable_should_search_searchable_custom_fields
+  def test_filter_any_searchable_no_matches
+    User.current = User.find(1)
+    query = IssueQuery.new(
+      :name => '_',
+      :filters => {
+        'any_searchable' => {
+          :operator => '~',
+          :values => ['SomethingThatDoesNotExist']
+        }
+      }
+    )
+    result = find_issues_with_query(query)
+    assert_empty result.map(&:id)
+  end
+
+  def test_filter_any_searchable_negative
+    User.current = User.find(1)
+    query = IssueQuery.new(
+      :name => '_',
```

```
+      :filters => {
+        'any_searchable' => {
+          :operator => '!~',
+          :values => ['recipe']
+        }
+      }
+    )
+    result = find_issues_with_query(query)
+    refute_includes [1, 2, 3], result.map(&:id)
+  end
+
+  def test_filter_any_searchable_negative_no_matches
+    User.current = User.find(1)
+    query = IssueQuery.new(
+      :name => '_',
+      :filters => {
+        'any_searchable' => {
+          :operator => '!~',
+          :values => ['SomethingThatDoesNotExist']
+        }
+      }
+    )
+    result = find_issues_with_query(query)
+    refute_empty result.map(&:id)
+  end
+
+  def test_filter_any_searchable_should_search_searchable_custom_fields
     User.current = User.find(1)
     query = IssueQuery.new(
       :name => '_',
```

#8 - 2023-04-05 15:01 - Go MAEDA

Holger Just wrote in #note-7:

> I was just having another look at r22164. Are you sure that the logic is correct for the !~ filter in case we find no issues with the search?
>
> If I'm reading the code correctly, we now restrict the final query result to also return no issues (by returning '1=0') in that case. However, I think we should probably not restrict the result at all in that case?

You are right, "doesn't contains" filter does not work correctly when no issue matches. Thank you for pointing it out.

#9 - 2023-04-05 15:53 - Go MAEDA

Committed the fix in #note-7 in r22168.

#10 - 2023-04-05 16:24 - Go MAEDA

I found another issue that should be fixed.

The new filter currently performs an OR search when multiple keywords are given, but I found that it should perform an AND search instead. This is because the "contains" operator for other string/text fields performs an AND search.

#11 - 2023-04-05 18:40 - Go MAEDA

- File any-searchable-text-should-perform-AND-search.patch added

Go MAEDA wrote in #note-10:

> I found another issue that should be fixed.
>
> The new filter currently performs an OR search when multiple keywords are given, but I found that it should perform an AND search instead. This is because the "contains" operator for other string/text fields performs an AND search.

This patch changes the behavior of the filter to AND search.

#12 - 2023-04-05 20:15 - Holger Just

Looks good to me, thanks!

#13 - 2023-04-06 01:37 - Go MAEDA

- Status changed from Reopened to Closed

- Resolution set to Fixed

Go MAEDA wrote in #note-11:

> Go MAEDA wrote in #note-10:
>
> > I found another issue that should be fixed.
> >
> > The new filter currently performs an OR search when multiple keywords are given, but I found that it should perform an AND search instead. This is because the "contains" operator for other string/text fields performs an AND search.
>
> This patch changes the behavior of the filter to AND search.

Committed this patch in r22169.

#14 - 2023-04-06 10:54 - Holger Just

I'd still have an updated German translation for the new filter if you like :)

```
diff --git a/config/locales/de.yml b/config/locales/de.yml
index 71c315e487..ad6e4c071d 100644
--- a/config/locales/de.yml
+++ b/config/locales/de.yml
@@ -280,6 +280,7 @@ de:
   field_active: Aktiv
   field_activity: Aktivität
   field_admin: Administrator
+  field_any_searchable: Durchsuchbarer Text
   field_assignable: Tickets können dieser Rolle zugewiesen werden
   field_assigned_to: Zugewiesen an
   field_assigned_to_role: Zuständigkeitsrolle
@@ -1439,4 +1440,3 @@ de:
     them is the better solution.
   text_users_bulk_destroy_confirm: To confirm, please enter "%{yes}" below.
   label_auto_watch_on_issue_created: Issues I created
-  field_any_searchable: Any searchable text
```

#15 - 2023-04-08 16:37 - Ivan Cenov

Good filter. Thanks.

#16 - 2023-04-09 08:46 - Go MAEDA

Holger Just wrote in #note-14:

> I'd still have an updated German translation for the new filter if you like :)

Thank you, I committed the update for German translation in r22174.

#17 - 2023-04-09 09:08 - Go MAEDA

- File clipboard-202304091605-ud44f.png added

- File or_search_for_any_searchable-text-filter.patch added

- Status changed from Closed to Reopened

Here is an additional patch for this feature. This patch adds a new operator "contains any of" (|~) to the "Any searchable text" filter.

The already implemented "contains" operator performs AND search. This new "contains any of" operator performs OR search.

clipboard-202304091605-ud44f.png

#18 - 2023-04-09 13:58 - Go MAEDA

- Has duplicate Feature #680: free text ticket filter added

#19 - 2023-04-09 17:40 - Go MAEDA

- Related to Feature #10574: Add filtering ability to search or search to filtering added

#20 - 2023-04-12 05:26 - Go MAEDA

- Status changed from Reopened to Closed

Go MAEDA wrote in #note-17:

> Here is an additional patch for this feature. This patch adds a new operator "contains any of" (|~) to the "Any searchable text" filter.
>
> The already implemented "contains" operator performs AND search. This new "contains any of" operator performs OR search.

I have posted a new patch that includes this patch (or_search_for_any_searchable-text-filter.patch):
Feature #38435: "contains any of" operator for text filters to perform OR search of multiple terms

#21 - 2023-04-18 11:10 - Go MAEDA

- File fix-any_searchable-does-not-support-mine-and-bookmarks.patch added

- Status changed from Closed to Reopened

The filter does not return any issue when used in combination with a project filter containing "my projects" or "my bookmarks". Attaching a patch to fix the issue.

#22 - 2023-04-18 12:52 - Holger Just

LGTM, thanks!

#23 - 2023-04-18 14:06 - Go MAEDA

- Status changed from Reopened to Closed

Go MAEDA wrote in #note-21:

> The filter does not return any issue when used in combination with a project filter containing "my projects" or "my bookmarks". Attaching a patch to fix the issue.

Committed the fix in r22203. Holger, thank you for quickly reviewing the patch!

#24 - 2024-06-20 11:33 - Go MAEDA

- Related to Feature #14943: Split the 'Status' search field added

#25 - 2024-06-20 11:39 - Go MAEDA

- Related to Feature #9180: Improve search system for issues - like "context specific search" added

Files

clipboard-202304021708-v4fbz.png	215 KB	2023-04-02	Go MAEDA
0001-Add-Any-serchable-text-filter-for-issues.patch	4.42 KB	2023-04-02	Go MAEDA
0002-Group-text-filters-in-the-filter-select-list.patch	1.9 KB	2023-04-02	Go MAEDA
0003-Reorder-groups-in-the-filter-select-list.patch	841 Bytes	2023-04-02	Go MAEDA
any-searchable-text-should-perform-AND-search.patch	1.9 KB	2023-04-05	Go MAEDA
clipboard-202304091605-ud44f.png	179 KB	2023-04-09	Go MAEDA
or_search_for_any_searchable-text-filter.patch	2.76 KB	2023-04-09	Go MAEDA
fix-any_searchable-does-not-support-mine-and-bookmarks.patch	2.37 KB	2023-04-18	Go MAEDA