## [API] Index 'like' queries and use for search

**04/17/2019 03:15 PM - Peter Amstutz**

<table>
<thead>
<tr>
<th>Status:</th>
<th>Resolved</th>
<th>Start date:</th>
<th>06/14/2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assigned To:</td>
<td>Eric Biagiotti</td>
<td>% Done:</td>
<td>100%</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td>2019-06-19 Sprint</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Description

Add a migration that creates trigram indexes on columns currently searchable by full text search. This should concatenate columns to create a single expression index (eg name || description || file_names).

Migration should execute "create extension pg_trgm" and provide a useful error message if the extension is not available (should advise the user to install postgres-contrib and/or link to the migration notes).

Update workbench2 search bar to use ["any", "like", "%term%"] instead of '@@'. Workbench should perform splitting of the search on whitespace and add a separate search clause for each term.

### Subtasks:

**Task # 15264: Review 15106-trgm-text-search** - Resolved

### Related issues:

Related to Arvados - Feature #14573: [Spike] [API] Fully functional filename ... - Resolved

### Associated revisions

**Revision 442a871e - 06/10/2019 02:07 PM - Eric Biagiotti**

Merge branch 'master' into 15106-trgm-text-search

refs #15106

Arvados-DCO-1.1-Signed-off-by: Eric Biagiotti <ebiagiotti@veritasgenetics.com>

**Revision b4b8e120 - 06/13/2019 05:22 PM - Eric Biagiotti**

Merge remote-tracking branch 'origin/master' into 15106-trgm-text-search

refs #15106

Arvados-DCO-1.1-Signed-off-by: Eric Biagiotti <ebiagiotti@veritasgenetics.com>

**Revision a7cdd1fa - 06/18/2019 02:06 PM - Eric Biagiotti**

Merge branch '15106-trgm-text-search'

refs #15106

Arvados-DCO-1.1-Signed-off-by: Eric Biagiotti <ebiagiotti@veritasgenetics.com>

**Revision fea9aae7 - 07/22/2019 08:13 PM - Eric Biagiotti**

Merge remote-tracking branch 'origin/15106-trgm-text-search'

refs #15106

Arvados-DCO-1.1-Signed-off-by: Eric Biagiotti <ebiagiotti@veritasgenetics.com>

### History

**#1 - 04/17/2019 03:15 PM - Peter Amstutz**
- Status changed from New to In Progress

**#2 - 04/17/2019 03:15 PM - Peter Amstutz**
- Subject changed from Indexed like queries to [API] Index 'like' queries for search
A few questions/comments before writing the migration:

- Looking at our full text search indexes, I noticed we are using COALESCE. I just wanted to double check whether this is needed for creating the trgm index.

- I also noticed from the examples on the spike that when creating the index, columns are separated by `||` `||`. I'm not sure what this does. Any explanation would be helpful.

- If I created the trgm index with the following:

```
create index collection_trgm_idx on public.collections USING gin ((file_names || ' ' || (properties)::text) public.gin_trgm_ops)
```

Then the following where clause is required to use the trgm index:

```
select * from collections where (file_names || ' ' || (properties)::text) ilike '%176DEMH%'
```

A regular query like the following will not use the trgm index:

```
select * from collections where file_names ilike '%176DEMH%'
```

I haven't dug into record_filter in full detail yet, but I'm not sure it will satisfy the above. For example, the "any" attribute uses the searchable_columns function, which doesn't appear to handle jsonb types. Could it be possible that we need another operator similar to @@, or am I way off base?

Eric Biagiotti wrote:

A few questions/comments before writing the migration:

- Looking at our full text search indexes, I noticed we are using COALESCE. I just wanted to double check whether this is needed for creating the trgm index.

I think that's because it isn't legal to concatenate a string with NULL. COALESCE specifies that if the column is null, use an empty string instead.
- I also noticed from the examples on the spike that when creating the index, columns are separated by `||` `||`. I'm not sure what this does. Any explanation would be helpful.

|| is the string concatenation operator. This is just adding a space between each column text.

- If I created the trgm index with the following:

```
cREATE INDEX collection_trgm_idx ON public.collections USING gin ((file_names || ' ' || (properties)::text) public.gin_trgm_ops)
```

Then the following where clause is required to use the trgm index:

```
SELECT * FROM collections WHERE (file_names || ' ' || (properties)::text) ilike '%176DEMH%'
```

A regular query like the following will not use the trgm index:

```
SELECT * FROM collections WHERE file_names ilike '%176DEMH%'
```

The trick here is that we want to build the index without actually creating and storing an additional column, so we make an "expression index". In order to use the index, you have to query on exactly the same expression.

I haven’t dug into record_filter in full detail yet, but I’m not sure it will satisfy the above. For example, the "any" attribute uses the searchable_columns function, which doesn’t appear to handle jsonb types. Could it be possible that we need another operator similar to @@, or am I way off base?

We’re only supporting searches on ‘any’. The record_filter code should be able to recognize ["any", "ilike", "%176DEMH%"] and build the appropriate query to use the trigram index.

You should be able to coerce jsonb columns to a string representation. It might be that you are doing it already with properties::text.

#13 - 05/31/2019 08:19 PM - Eric Biagiotti
Some migrations statistics using the e51c5 data on my laptop.

<table>
<thead>
<tr>
<th>Table Name</th>
<th># of Rows</th>
<th>Index Creation Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>collections</td>
<td>4228696</td>
<td>575.6284</td>
</tr>
<tr>
<td>container_requests</td>
<td>873201</td>
<td>219.8006</td>
</tr>
<tr>
<td>groups</td>
<td>1789</td>
<td>0.1482</td>
</tr>
<tr>
<td>jobs</td>
<td>359324</td>
<td>134.3642</td>
</tr>
<tr>
<td>pipeline_instances</td>
<td>10721</td>
<td>31.8514</td>
</tr>
<tr>
<td>pipeline_templates</td>
<td>6</td>
<td>0.0074</td>
</tr>
<tr>
<td>workflows</td>
<td>19</td>
<td>0.0024</td>
</tr>
</tbody>
</table>

Total Migration Time: ~ 16 minutes

#14 - 06/04/2019 05:44 PM - Eric Biagiotti
Some questions/comments about needing to be a postgres superuser to execute "CREATE EXTENSION IF NOT EXISTS pg_trgm". This prevents us from creating trgm indexes in a database migration unless pg_trgm is already enabled or we can temporarily grant superuser permissions in the migration. 3 use cases to consider: run-tests, arvbox, and deployment.

- This project ([https://github.com/dimitri/pgextwlist](https://github.com/dimitri/pgextwlist)) is meant to make it possible to add extensions as a non-super user. Seems straightforward to add, but I’d want to get input before going down this road. Not sure of potential issues with deployment, etc.

- For deployment, if we use a post-install script to enable the extension, it would have to run before db:migrate but after db:create.

- We could use a template database for run-tests/arvbox or any instance when creating the database for the first time.

- Could we create a new rake task to slide in between db:create and db:migrate? Can this be run as a super user?

#15 - 06/05/2019 01:15 PM - Eric Biagiotti
- Target version changed from 2019-06-05 Sprint to 2019-06-19 Sprint

#16 - 06/05/2019 01:18 PM - Eric Biagiotti
- Status changed from New to In Progress
Added full_text_trgm function to ArvadosModel to create the string required for creating the trgm index and querying with it. Uses much of the same code from full_text_tsvector, but I didn't change that function since its used in other migrations.

Added trigram search to the api for ["any", "ilike", %example%].

Added test to make sure trigram indexes exist on the right tables/column names.

Updated arvbox to install postgresql-contrib and create a superuser

Updated https://dev.arvados.org/projects/arvados/wiki/Hacking_prerequisites to include postgresql-contrib/superuser instructions

Added a note to the upgrade docs and updated the install docs to include the necessary steps for the migration to run.


Some performance metrics from running arv collection list --filter on 3049 collections returning 80 results:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Filter</th>
<th>Resulting Query</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigram</td>
<td>[&quot;any&quot;, &quot;ilike&quot;, &quot;%test%&quot;]</td>
<td>SELECT ... FROM collections WHERE ((coalesce(owner_uuid,&quot;&quot;)</td>
<td></td>
</tr>
<tr>
<td>No Trigram</td>
<td>[&quot;any&quot;, &quot;ilike&quot;, &quot;%test%&quot;]</td>
<td>SELECT ... FROM collections WHERE ((owner_uuid ilike 'test%' OR current_version_uuid ilike 'test%'))</td>
<td>17.8 ms</td>
</tr>
<tr>
<td>Full Text Search</td>
<td>[&quot;any&quot;, &quot;@@&quot;, &quot;%test%&quot;]</td>
<td>SELECT ... FROM collections WHERE ((coalesce(owner_uuid,&quot;&quot;)</td>
<td></td>
</tr>
</tbody>
</table>

Anything special I need to do for packaging? I built and tested packaging locally and it worked fine, but I noticed the centos7 docker file installs some postgres dependencies.

#19 - 06/17/2019 05:06 PM - Peter Amstutz

if attrs_in == 'any' && operator.casemcmp('ilike') && operand.is_a? String) && operand.match("^[%].*[%]$")

• Should include "like" (regular case-sensitive match) as well as "ilike" (case-insensitive).

• What happens if the user request like "any like foo" so operand.match("^[%].*[%]" doesn't apply? It looks like it will just be ignored? Should that be an error?

#20 - 06/17/2019 06:05 PM - Eric Biagiotti

Peter Amstutz wrote:

[...]

• Should include "like" (regular case-sensitive match) as well as "ilike" (case-insensitive).

Fixed in 0719567935227976b0331c9fd231a184bcabd1a
What happens if the user request like "any like foo" so operand.match('^[\%].*[^\%]$') doesn't apply? It looks like it will just be ignored? Should that be an error?

In this case, it falls through and model_class.searchable_columns (In 139) is used to construct a query similar to row 2 in the table from note 18.

I was under the assumption that we only wanted to use the trigram index with operands that begin and end with '%'. Note that the trigram indexes are built using ArvadosModel.full_text_searchable_columns, which is not necessarily a superset of ArvadosModel.searchable_columns, which is what 'like/\like' are using by default.

Eric Biagiotti wrote:

> What happens if the user request like "any like foo" so operand.match('^[\%].*[^\%]$') doesn't apply? It looks like it will just be ignored? Should that be an error?

In this case, it falls through and model_class.searchable_columns (In 139) is used to construct a query similar to row 2 in the table from note 18.

Got it. That will do a full table scan but at least it will do something reasonable.

I was under the assumption that we only wanted to use the trigram index with operands that begin and end with '%'. Note that the trigram indexes are built using ArvadosModel.full_text_searchable_columns, which is not necessarily a superset of ArvadosModel.searchable_columns, which is what 'like/\like' are using by default.

Yes, since they are all combined into a single string that makes sense. If we need to index for leading/trailing matches we can index individual columns.

Eric Biagiotti wrote:

> (good catch adding .zero? to casecmp)

I kicked off tests:

[link to test results]

- Status changed from In Progress to Resolved