# Arvados - Bug #4523

## [Workbench] Search dialog giving error when searching in "All projects" in qr1hi

11/14/2014 01:40 AM - Radhika Chippada

| | | | |
|---|---|---|---|
| **Status:** | Resolved | **Start date:** | 12/08/2014 |
| **Priority:** | Normal | **Due date:** | |
| **Assigned To:** | Radhika Chippada | **% Done:** | 100% |
| **Category:** | Workbench | **Estimated time:** | 0.00 hour |
| **Target version:** | 2015-01-28 Sprint | | |

**Description**

To reproduce:

Go to https://workbench.qr1hi.arvadosapi.com/, enter any term into the search box (I used "hash") and click the search button. The search modal box will appear and display a spinning wheel. After some time it issues "Oops, request failed."

Proposed fix:

- Add database indexes to all tables:
  - one on owner_uuid
  - one on the full set of searchable columns (i.e., ModelClass.searchable_columns("<") | ModelClass.searchable_columns("like"))
  - (phase2) each searchable column
- To avoid undetected additions of columns without indexes in the future, add a unit test that iterates over all model classes and tests:
  - there is a multi-column index on the full set of searchable columns
  - there is an index on owner_uuid and an index on uuid
  - (phase2) there is an index on each searchable column

Fix phase 3 (see subtask #4900): We want "search for filename" to be fast, but manifest_text is too big to index. Therefore we want to add a new column of type string with file names only, truncated to the allowed size.

- Add a file_names column to collections table (:string, length: 2**16).
- Add a before_save hook to update the file_names attribute whenever manifest_text changes. If a manifest has more than 64KiB worth of filenames, just truncate the file_names attribute. (Saving the manifest will still work, but some filenames won't be searchable.)
  - Example manifest_text: ". d41d8[...] 0:0:file1.txt 0:0:file2.txt\n"
  - Example file_names: "file1.txt\nfile2.txt\n"
- Use file_names instead of manifest_text when when searching for "any" in collections table.
- Omit text columns from searchable_columns.
- Add tests:
  - file_names content is correct after create, and after update.
  - Searching by filename works.
  - Using a manifest with 1MiB worth of filenames (which can be built reasonably quickly like this)

```
. d41d8[...] 0:0:longlonglongfilename0000001 0:0:longlonglongfilename0000002 ...\n
```

    - Creating a collection with this manifest does not throw an error
    - A search for longlonglongfilename0000001 includes the collection that was just created
  - The collection API response (create/update/get/index) does not provide a file_names attribute.

**Subtasks:**

| | |
|---|---|
| Task # 4737: Add index on owner_uuid column for all data models that include the owner_... | **Resolved** |
| Task # 4740: Add search index on all data models | **Resolved** |
| Task # 4900: Add a "file_names" column for collections data model | **Resolved** |
| Task # 4746: Review branch: 4523-search-index | **Resolved** |

**Related issues:**

| | | |
|---|---|---|
| Related to Arvados - Feature #4024: [Workbench] PipelineInstances#index shoul... | **Resolved** | **10/28/2014** |
| Related to Arvados - Bug #4942: [Workbench] Data Collections Tab in Projects ... | **Closed** | **01/08/2015** |
| Related to Arvados - Bug #4464: [Workbench] Collections tab loads forever on ... | **Resolved** | **02/05/2015** |

## Associated revisions

**Revision 1e31ae09 - 12/29/2014 08:12 PM - Tom Clegg**

Merge branch '4523-owner_uuid-index' refs #4523

**Revision cadb785f - 01/14/2015 08:34 PM - Radhika Chippada**

closes #4523
Merge branch '4523-search-index'

**Revision 212e68e9 - 01/14/2015 10:33 PM - Tom Clegg**

Reduce file_names limit to 2^12. refs #4523.

**Revision 1fae5d56 - 01/22/2015 05:06 PM - Radhika Chippada**

refs #4523
Merge branch '4523-search-index'

**Revision 643bf501 - 01/27/2015 06:43 PM - Radhika Chippada**

refs #4523
Merge branch '4523-search-index'

**Revision 9b6b5f0b - 02/10/2015 03:58 PM - Radhika Chippada**

refs #4523 : Add postgres full text search support
Merge branch '4523-full-text-search'

## History

**#1 - 11/14/2014 01:42 AM - Radhika Chippada**

*- Subject changed from [Workbench] Search dialog giving error in qr1hi to [Workbench] Search dialog giving error when searching in "All projects" in qr1hi*

**#2 - 11/14/2014 04:34 PM - Tim Pierce**

*- Description updated*

**#3 - 11/14/2014 04:34 PM - Tim Pierce**

*- Category set to Workbench*

**#4 - 11/14/2014 04:35 PM - Tim Pierce**

*- Description updated*

**#5 - 11/14/2014 08:10 PM - Tom Clegg**

*- Status changed from New to In Progress*

*- Story points set to 0.5*

**#6 - 11/14/2014 08:10 PM - Tom Clegg**

*- Target version changed from Bug Triage to 2014-11-19 sprint*

**#7 - 11/17/2014 05:39 PM - Radhika Chippada**

$ #<ArvadosApiClient::NoApiResponseException: HTTPClient::ReceiveTimeoutError error connecting to API server>
/home/radhika/arvados/apps/workbench/app/models/arvados_api_client.rb:133:in `rescue in block in api'
/home/radhika/arvados/apps/workbench/app/models/arvados_api_client.rb:130:in `block in api'
/home/radhika/arvados/apps/workbench/app/models/arvados_api_client.rb:129:in `synchronize'
/home/radhika/arvados/apps/workbench/app/models/arvados_api_client.rb:129:in `api'
/home/radhika/arvados/apps/workbench/app/models/group.rb:11:in `contents'
/home/radhika/arvados/apps/workbench/app/controllers/search_controller.rb:16:in `find_objects_for_index'
/home/radhika/arvados/apps/workbench/app/controllers/application_controller.rb:271:in `block (2 levels) in choose'

**#8 - 11/19/2014 08:18 PM - Tom Clegg**

*- Assigned To set to Tom Clegg*

**#9 - 11/19/2014 08:19 PM - Tom Clegg**

*- Target version changed from 2014-11-19 sprint to 2014-12-10 sprint*

#### #10 - 12/05/2014 07:23 PM - Tom Clegg

I *think* "ActiveRecord: 51 seconds" means we need better database indexes:

```
Started GET "/arvados/v1/groups/contents" for 127.0.0.1 at 2014-12-05 19:17:02 +0000
Processing by Arvados::V1::GroupsController#contents as */*
  Parameters: {"api_token"=>"65aky3o8lf667tllxfjeemauowho5gboz7s0btzangcn6yvggr", "reader_tokens"=>"[]", "limi
t"=>"200", "offset"=>"0", "filters"=>"[[\"any\",\"ilike\",\"%picard%\"]]", "include_linked"=>"1"}
...
Completed 200 OK in 52642.3ms (Views: 390.7ms | ActiveRecord: 51386.0ms)
```

We should *at least* tell Postgres to index the columns that we search when a client uses an ['any',...] filter!

#### #11 - 12/05/2014 07:38 PM - Tom Clegg

*- Description updated*

#### #12 - 12/05/2014 07:40 PM - Tom Clegg

*- Assigned To changed from Tom Clegg to Radhika Chippada*

#### #13 - 12/05/2014 07:42 PM - Tom Clegg

*- Description updated*

#### #14 - 12/05/2014 07:43 PM - Tom Clegg

*- Description updated*

#### #15 - 12/05/2014 07:51 PM - Tom Clegg

*- Description updated*

#### #16 - 12/05/2014 07:53 PM - Tom Clegg

*- Description updated*

#### #17 - 12/08/2014 06:15 PM - Radhika Chippada

4523-owner_uuid-index includes:

- Migration script to add owner_uuid index on all tables that have this column
- Add test to check the existence of owner_uuid index on all such tables
- Also, add test to check uuid unique index exists on all tables that have the uuid column. Currently, all tables have this and hence no additional db migration is needed for this.

#### #18 - 12/08/2014 09:06 PM - Radhika Chippada

4523-search-index includes:

- changes included in branch 4523-owner_uuid-index
- search index for all the data models that support the searchable_columns method
- test to verify that all those tables have the search index (after doing the db migration)

#### #19 - 12/08/2014 10:36 PM - Brett Smith

Reviewing a63c263

- This story is going to need some revision—at least one exception—because the Logs table index can't handle all the data in log entries. I got these failures running the Workbench tests:

```
  1) Error:
WebsocketTest#test_test_live_logging_scrolling_jobs:
ArvadosApiClient::ApiErrorResponseException: #<ActiveRecord::StatementInvalid: PG::ProgramLimitExceeded: E
RROR:  index row size 3408 exceeds maximum 2712 for index "logs_search_index"
HINT:  Values larger than 1/3 of a buffer page cannot be indexed.
Consider a function index of an MD5 hash of the value, or use full text indexing.
: INSERT INTO "logs" ("created_at", "event_at", "event_type", "modified_at", "modified_by_client_uuid", "m
odified_by_user_uuid", "object_owner_uuid", "object_uuid", "owner_uuid", "properties", "summary", "updated
_at", "uuid") VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13) RETURNING "id"> [API: 422]
    app/models/arvados_api_client.rb:148:in `api'
```

```
    test/integration/websockets_test.rb:56:in `block (2 levels) in <class:WebsocketTest>'
    test/test_helper.rb:256:in `run'

  2) Error:
WebsocketTest#test_test_live_logging_scrolling_pipeline_instances:
ArvadosApiClient::ApiErrorResponseException: #<ActiveRecord::StatementInvalid: PG::ProgramLimitExceeded: E
RROR:  index row size 3408 exceeds maximum 2712 for index "logs_search_index"
HINT:  Values larger than 1/3 of a buffer page cannot be indexed.
Consider a function index of an MD5 hash of the value, or use full text indexing.
: INSERT INTO "logs" ("created_at", "event_at", "event_type", "modified_at", "modified_by_client_uuid", "m
odified_by_user_uuid", "object_owner_uuid", "object_uuid", "owner_uuid", "properties", "summary", "updated
_at", "uuid") VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13) RETURNING "id"> [API: 422]
    app/models/arvados_api_client.rb:148:in `api'
    test/integration/websockets_test.rb:56:in `block (2 levels) in <class:WebsocketTest>'
    test/test_helper.rb:256:in `run'
```

- 20141208185217_search_index.rb calls searchable_columns on model classes. This means that the behavior of the migration will change as the behavior of the models change. This could cause the migration to fail in future codebases if, e.g., we add a new searchable column and the migration to make it searchable simultaneously. I think you need to hardcode the columns that are currently searchable in the migration itself.
- The copy of structure.sql in this branch is not up-to-date with the included migrations. It only includes the search indexes, and not the ones for owner_uuid. It also doesn't increment schema_migrations.
- There's trailing whitespace in arvados_model_test.rb.

Thanks.

#### #20 - 12/09/2014 02:35 PM - Brett Smith

Brett Smith wrote:

- This story is going to need some revision—at least one exception—because the Logs table index can't handle all the data in log entries.

Reflecting on it some more, I think I would suggest that we should only make indexes for columns that are strings or smaller. Not larger text fields.

#### #21 - 12/09/2014 02:49 PM - Tom Clegg

Brett Smith wrote:

> Reflecting on it some more, I think I would suggest that we should only make indexes for columns that are strings or smaller. Not larger text fields.

Postgres suggests we consider full text indexing, which seems reasonable. Would that mean changing our queries to use full text operators instead of "like" etc.?

The bug at hand is that the site-wide search is so slow it times out. We should be able to fix this without losing the ability to match filenames in collections, although if I had to choose one or the other I'd choose "no timeout." If I'm following, that would mean removing "text" fields from searchable_columns (and from the indexes).

#### #22 - 12/09/2014 07:29 PM - Radhika Chippada

Notes about latest:

- Updated api arvados_model searchable_columns method to omit text type columns when using "ilike" operator. This now results in not including / expecting a text type column in search index.
  - Note: I needed to limit this exclusion on ilike operator only. Omitting the text type columns from searchable_columns altogether results in omitting these when operator is "like" and this causes system-wide errors with all other where clause related lookups.
- Updated the migration scripts to hard code tables and corresponding search columns.
- The updated db/structure.rb should include both migration script related info. Please retest.

Questions:

- Do we want to omit "owner_uuid", "modified_by_client_uuid", "modified_by_user_uuid" in search indexes? I am not sure how these are useful in a search query.

#### #23 - 12/10/2014 08:22 PM - Ward Vandewege

*- Target version changed from 2014-12-10 sprint to 2015-01-07 sprint*

#### #24 - 12/12/2014 09:08 PM - Tom Clegg

(From IRC) Changing description from "text" to "string" seems like a good idea, provided it's no longer true that Rails doesn't know how to do varchar. We'd probably want description to handle at least, say, 10K. (AFAICT there is no great reason to use a smaller limit.)

#### #25 - 12/13/2014 08:29 PM - Radhika Chippada

Tom, pushed an update to the 4523-search-index branch with description column updated from text to string of length 10000 on groups, jobs, pipeline_instances, and pipeline_templates. In addition, updated description column on collections to size 10000. These 5 are the only tables with description column as of now. Also, updated the search index migration script to include description column. Updated the test accordingly as well. Please take another look. Thanks.

#### #26 - 01/05/2015 11:39 PM - Tom Clegg

*- Description updated*

#### #27 - 01/06/2015 05:29 AM - Tom Clegg

*- Description updated*

#### #28 - 01/07/2015 08:46 PM - Radhika Chippada

*- Target version changed from 2015-01-07 sprint to 2015-01-28 Sprint*

#### #29 - 01/08/2015 04:01 PM - Tom Clegg

At cf67055

Workbench: Good catch. But I wonder whether/how we should communicate this distinction in the discovery doc, now that type=string/text doesn't tell the client.

With the change to arvados_model.rb, clients are not allowed to do an "ilike" query on text fields, but they're allowed to do a "like" query, which suffers from the same indexing problem. This case should be either true or false, it shouldn't depend on the operator. I'd prefer to make it false, and remove the "# Disappointing, but text columns aren't indexed yet." loophole in the test case.

In Collection#set_file_names:

- Use "\n" instead of space.
- Convert \040 to space in filenames.
- Skip block tokens.
- Avoid using the same name for a local variable and an instance variable!
- For a large manifest, self.manifest_text.split will create a large array. We can make this more efficient (and avoid block hash tokens) by using the block form of scan:

- ```
  [redacted].file_names = ''
  if manifest_text
    manifest_text.scan(/ \d+:\d+:(\S+)/) do |file_name|
      file_names << file_name.gsub('\040',' ') + "\n"
      break if file_names.length > 2**13
    end
  end
  ```

  (this doesn't do uniq(), though)
- Also include stream names. (Perhaps a separate loop?)

Migration:

- Use the real Collection#file_names method, not a copy of it. (Perhaps split set_file_names into one method that always updates, and one that calls it if manifest_text_changed?, so you can use the former in the migration.)

In the test case for manifest/file_names:

- It takes 16 seconds to make each 256K string on my system. It should be much faster if it loops over s+=s instead of s+="a".
- The "description size" tests are independent. They should be in their own test case(s) instead of being bundled with the manifest/file_names tests.
- If a 2**18-byte description does *not* raise an exception, that's good news, not bad news! I think what we want here is a functional test: if a long description is sent by a client, either it works *or* it produces a useful error message.

The "create collection, verify file_names not returned, and search with filename" test has a lot of stuff in it, most of which seems unrelated to what we say we're testing (e.g., assert_nil json_response['description']). I think it should be refactored into multiple tests which are very focused, like:

- Integration test creates a collection and finds it using ['any','like','%filename%'].
- Functional test ensures GET /collections/fixture-uuid does not provide a file_names attr.

#### #30 - 01/09/2015 06:59 PM - Radhika Chippada
  ```

Tomclegg:

Some questions and comments about your comments.

Question 1:

> With the change to arvados_model.rb, clients are not allowed to do an "ilike" query on text fields, but they're allowed to do a "like" query, which suffers from the same indexing problem. This case should be either true or false, it shouldn't depend on the operator. I'd prefer to make it false, and remove the "# Disappointing, but text columns aren't indexed yet." loophole in the test case.

It appears that we may not be able to remove "like" support. If you look in workbench -> app/views/collections/_show_source_summary.html.erb, line 7 uses the following to find out which pipelines used a collection:

```
pipelines = PipelineInstance.filter([["components", "like", "%#{@object.uuid}%"]]).each do |pipeline|
```

It does not appear that we have any other means of finding out an answer to this question if we were to stop supporting "like". I think we hence need to limit this to ilike only which is what is used by search. Please let me know what you think.

Question 2:

> Workbench: Good catch. But I wonder whether/how we should communicate this distinction in the discovery doc, now that type=string/text doesn't tell the client.

Please clarify what we want to do with this.

Question 3:

> Also include stream names. (Perhaps a separate loop?)

Please let me know how I look for streams, such as the pattern to look for. I am currently not at all clear how this can be handled.

#### #31 - 01/09/2015 08:58 PM - Tom Clegg

Radhika Chippada wrote:

> It does not appear that we have any other means of finding out an answer to this question if we were to stop supporting "like". I think we hence need to limit this to ilike only which is what is used by search. Please let me know what you think.

OK. I don't want to break that feature. But I still don't think it's correct to treat like and ilike differently in searchable_columns. That seems to serve only to fool the "everything searchable has an index" test into passing when it should be failing. So it should just go back to

```
when :string, :text
  true
```

How about instead we override searchable_columns in the Collection class with super - [:manifest_text] (pipeline searches will still be slow, but collection searches will be OK)?

> Question 2:
>
> > Workbench: Good catch. But I wonder whether/how we should communicate this distinction in the discovery doc, now that type=string/text doesn't tell the client.
>
> Please clarify what we want to do with this.

For now, definitely nothing.

(Also, perhaps nothing ever. Thinking again, this looks like a UI decision that Workbench just has to figure out by itself, as you've done.)

> Question 3:
>
> > Also include stream names. (Perhaps a separate loop?)
>
> Please let me know how I look for streams, such as the pattern to look for. I am currently not at all clear how this can be handled.

I think that would be

```
...
manifest_text.scan(/^\.\/(\S+)/m) do |stream_name|
  file_names << stream_name.gsub('\040',' ') + "\n"
  break if file_names.length > 2**13
end
...
```

#### #32 - 01/10/2015 02:46 AM - Radhika Chippada

Tom, thanks for clarifying about what streams are.

I made the change (for the time being) to allow text attributes in search for non-collection objects (by overriding searchable_columns in collections).

However, that this is not going to be acceptable for us. By allowing text typed attributes, especially from pipleline_instances - of which there are a lot in production, we will continue to see Timeout errors for searches from search dialog.

The operator 'like' is used by workbench for other non-search-dialog based searches and for reasons such as _show_source_summary.html.erb, we do not want to un-support it. However, in order to make search dialog work, we have to omit all text typed attributes for the ilike operator.

I do not know why you initially picked ilike for search-dialog implementation. However, since we are only using ilike for search-dialog based searches, we need to focus on fixing this so that it works as intended.

The right approach, in my opinion, would be to use ilike for searches such as search-dialog and like for others such as show_source_summary.html.erb and hence I do not see any harm by having two different implementations for like and ilike.

Let me know your thoughts. Thanks.

#### #33 - 01/13/2015 04:45 PM - Tom Clegg

Radhika Chippada wrote:

> Tom, thanks for clarifying about what streams are.
>
> I made the change (for the time being) to allow text attributes in search for non-collection objects (by overriding searchable_columns in collections).
>
> However, that this is not going to be acceptable for us. By allowing text typed attributes, especially from pipleline_instances - of which there are a lot in production, we will continue to see Timeout errors for searches from search dialog.
>
> The operator 'like' is used by workbench for other non-search-dialog based searches and for reasons such as _show_source_summary.html.erb, we do not want to un-support it. However, in order to make search dialog work, we have to omit all text typed attributes for the ilike operator.
>
> I do not know why you initially picked ilike for search-dialog implementation. However, since we are only using ilike for search-dialog based searches, we need to focus on fixing this so that it works as intended.

"ilike" is case-insensitive. "like" is case-sensitive. For more predictable behavior, Workbench should pick one and use it everywhere. (Workbench also uses regexps in some places.)

> The right approach, in my opinion, would be to use ilike for searches such as search-dialog and like for others such as show_source_summary.html.erb and hence I do not see any harm by having two different implementations for like and ilike.

I don't like this because it makes two different decisions with the operator.

| Filter | Behavior |
|---|---|
| ['any','like','%foo%'] | Get records where any string/text field contains "foo" |
| ['any','ilike','%foo%'] | Get records where any string field contains "foo", "Foo", etc. Ignore text fields. |

The intent of the "searchable depends on operator" pattern here is to transparently skip comparisons that don't make sense, so a boolean can participate in "any == true" without crashing on "any like foo".

If we want to support two kinds of search -- a slow one and a fast one -- then the "any" attribute selector seems like a better place to do that, rather than using two different operator types. (Surely we don't want ['any','i<','foo']...)

However, I don't think we really *want* to support "slow-and-useful" and "fast-but-not-useful-enough" searches. I think we just want to make the useful searches faster. For now, it's OK that some searches are slow: even "slow, sometimes maybe so slow it's impossible" is more useful than "cannot ever find that thing from this part of Workbench". (Especially if it's the Search dialog!)

Looking at a66a265...

Some extra whitespace at EOL in source:services/api/app/models/collection.rb

I think manifest_files should be an instance method, not a class method:

- def self.manifest_files manifest_text → def manifest_file_names
- self.file_names = Collection.manifest_files self.manifest_text → self.file_names = manifest_file_names
- file_names = Collection.manifest_files c.manifest_text → file_names = c.manifest_files (in migration)

In the migration:

- use parameterized SQL query instead of '#{file_names}'. (What if file_names contains an apostrophe?)
- use the Collection.find_each(batch_size: 20) do ... form to avoid retrieving the entire database table at once. Collections can be huge!
- Remove the if c.manifest_text condition, so the results from the migration are the same as if they had been generated during save.

#### #34 - 01/14/2015 07:38 AM - Tom Clegg

*- Story points changed from 0.5 to 2.0*

Thanks, I think 554728c is good to merge.

I notice the migration shows the entire list of files in the terminal, because we're using update_sql. Can you please check with Ward about whether this unusual amount of "rake db:migrate" output has potential to cause trouble when deploying?

I see a couple of testing issues I think we should address before closing the ticket, too:

1. The unit tests should check that c.file_names actually has (at least one of) the expected filenames, not just that c.file_names is not null. Even an empty string would pass those "very large manifest" tests...
   - assert_match might be the most appropriate tool here: exactly how it's encoded isn't vital, as long as it has the filenames. It would also make the default assertion failure message more helpful than "expected null to be true".
   - The unit tests should also confirm that stream names are searchable (use "blurfl" instead of ".", and assert_match it).
2. Clean up the integration tests a bit more
   - Remove extra assertions. For example, in "create collection, update description, and search for description", I don't think we actually want to test that portable_data_hash is 087e2f211aba2881c08fd8d1646522a3+51, do we? If that assertion fails (but the test cases specifically aimed at portable_data_hashes pass), the appropriate fix would be to update the test case. And if it fails along with a sea of other assertions throughout the test suite, it merely makes it a little harder to see what broke by looking at the test report. In either case the assertion seems to create work for us, without helping us notice/find bugs.
   - Instead of copying & pasting the manifest-signing logic, either (1) use an existing fixture instead of creating a new collection (fastest!), (2) call Collection.sign_manifest(". [...]\n", api_token(:active)), or (3) use an admin token so you don't need to sign the locators at all.
3. One other minor point I noticed earlier but then forgot to mention (sorry): I think it'd be better to use {filters:[['any','ilike',search_filter]]} in the tests instead of {where:{any:['contains',search_filter]}}, just because filters is the  documented/supported way.

#### #35 - 01/14/2015 03:13 PM - Tom Clegg

I noticed db/structure.sql has mixed up the columns of pipeline_instances again. The correct column order is whatever you get by doing all of the migrations in order. Try this:

```
# Down-migrate to August
RAILS_ENV=test bundle exec rake db:migrate VERSION=20140817035914

# Apply migrations in given order
RAILS_ENV=test bundle exec rake db:migrate

# Is db/structure.sql correct?
git diff
```

When I do the above on 554728c, I get the following diff.

```
--- a/services/api/db/structure.sql
+++ b/services/api/db/structure.sql
@@ -684,11 +684,11 @@ CREATE TABLE pipeline_instances (
     components text,
     updated_at timestamp without time zone NOT NULL,
     properties text,
-    state character varying(255),
     components_summary text,
-    description character varying(524288),
+    state character varying(255),
     started_at timestamp without time zone,
-    finished_at timestamp without time zone
+    finished_at timestamp without time zone,
+    description character varying(524288)
 );
```

Column order has gone back and forth in 6137770, e350e7a, a8c9797, 155e542, 8714664, 7d6b0f3... If I'm reading the diffs correctly, it looks like your development DB has the wrong column order, which you should be able to fix by doing the above down+up migration using RAILS_ENV=development.

#### #36 - 01/14/2015 06:55 PM - Tom Clegg

At 57f1fd5 I see two of these:

```
    signed_locator = Collection.sign_manifest("0:44:my_updated_test_file.txt\n", api_token(:active))
```

I expect this to return "0:44:my_updated_test_file.txt\n". I suppose it doesn't make the test fail, but it does produce a manifest here that doesn't make a lot of sense and could reasonably be rejected by future stricter manifest validations. I suspect it's not what you intended.

```
collection: "{\"manifest_text\":\". #{signed_locator} 0:44:my_updated_test_file.txt\\n\"}"
# => ". 0:44:my_updated_test_file.txt 0:44:my_updated_test_file.txt\n"
```

...amiright?

The change to db/structure.sql looks correct, thanks. Apparently *my* database was the one that was wrong about putting components_summary before state, but this migration was earlier than the irreversible collections migration so I didn't go back far enough to catch it. Fixed now!

#### #37 - 01/14/2015 08:40 PM - Radhika Chippada

*- Status changed from In Progress to Resolved*

*- % Done changed from 75 to 100*

Applied in changeset arvados|commit:cadb785fc63280862d71376def0128e4c70951f0.

#### #38 - 01/20/2015 04:21 PM - Tom Clegg

Looking at 1b01104 -- this looks promising.

I think we should index the file_names column rather than the raw manifest (block hashes aren't really amenable to full-text search, are they? and it would be good if "foo bar" could match a filename "foo bar.txt", which would look like "foo\040bar" in a manifest...)

Full-text search seems like something you usually want to do on all columns at once. Postgres seems to have features for weighting columns differently, ranking results, etc. I suppose the obvious API would be

```
{"filters":[["any","@@","foo bar"]]}
```

This would invoke plainto_tsquery() which would interpret "foo bar" as "foo & bar". We could add support for explicit operators -- perhaps triggered by presence of " & " or " | " -- but I think "let postgres parse the query" is a good thing to try first.

The docs suggest stuff like this:

```
to_tsvector('english', coalesce(title,'') || ' ' || coalesce(body,''));
```

I think we'll need to take steps to ensure the to_tsvector in our query is the same as the to_tsvector in our index. Perhaps in this case Collection could have a class method fulltext_tsvector that returns the "to_tsvector(...)" string; the migration and the query could both call that when building the SQL query; and a test case could verify that the database has an index on that tsvector (to ensure the database doesn't fall out of sync with the class method).

In the case of Collection we probably want something like this?

```
setweight(to_tsvector(coalesce(name,'')), 'A')     ||
setweight(to_tsvector(coalesce(description,'')), 'B')  ||
setweight(to_tsvector(coalesce(file_names,'')), 'C') ||
setweight(to_tsvector(coalesce(properties,'')), 'D')
```

The docs also suggest that performance is better if we add a column to hold the tsvector, but that it's "more important when using [the kind of index we're not using]". So maybe we should skip it for now? It seems like it would be easy enough to add later if it seems necessary.

#### #39 - 01/22/2015 02:07 PM - Radhika Chippada

Tom, please take another look at the 4523-full-text-search branch.

What is new:

- Updated the full-text migration script to add the full text search for all columns for the project contents object types (collections, groups, jobs, pipeline_instances, pipeline_templates). In case of collections, omitted manifest_text from the full text search. Ignored the "other" object types (humans, specimens, and traits from the indexing)
- Continued to use to_tsquery with logic included to support prefix matching. Otherwise, only full words matching the entered search filter are fetched.

```
  cond_out << model_class.full_text_tsvector+" @@ to_tsquery(?)"
  param_out << operand.split.each {|s| s.concat(':*')}.join(' & ')
```

- We can support '&' or '|' in full text search if entered by user (no space between words and exact matches only)

```
foo&bar  -- works; but looks for full words foo & bar only, not as prefixes
foo|bar  -- works; but looks for full words foo | bar only, not as prefixes

foo & bar  -- does not work
foo | bar  -- does not work
```

- I updated record_filters to support the full-text search operator '@@' and add unit and integration tests to verify that search filter works

```
    get '/arvados/v1/collections', {
      :filters => [['any', '@@', 'foo']].to_json
    }, auth(:active)
```

Questions / TBD:

- What do I need to update to use full-text search when search dialog is used in workbench?
- Workbench currently allows "empty" search string, that is, a user can click the top nav search icon with no text entered in search box. Currently, we fetch the "All projects" contents and display. However, full text search does not work with empty string. I think we need to not support this use case anymore?
- I have not yet added the setweight for the full text search columns.
- Cleaning up the "other" search index or the current searchable_columns etc.

Thanks.

#### #40 - 01/23/2015 08:29 PM - Radhika Chippada

It appears that we need the tsvector column to use setweight for ranking.

http://www.sai.msu.su/~megera/postgres/fts/doc/fts-complete-tut.html

#### #41 - 01/27/2015 03:18 PM - Tom Clegg

At 68bc991 on 4523-search-index, just a couple of minor suggestions:

- Instead of index_columns.select.each do ..., just need index_columns.select do ...
- change() runs for both up and down. I think this method should be up, and down should be empty. (If everything is working correctly, it's academic, because it's a no-op after it's run once.)
- has_description sounds like a boolean, but isn't. As a ruby exercise, I'd propose

```
has_description = index_columns.map(&:column).find_index('description')
if has_description
  ...migrate...
```

- The andand seems superfluous: if search_index_by_name.first is nil, then andand gets us past that line, but then index_columns is nil and the very next line crashes on nil.select. If the reason for this is "do nothing (instead of crashing) if the earlier migration hasn't run at all (although that would be surprising)" how about this:

```
break if search_index_by_name.empty?
```

- Even though the git commit message already says what's up, how about a comment in the migration itself too:

```
# If the database reflects an obsolete version of the 20141208185217
# migration (i.e., before commit:5c1db683), revert it and reapply the
# current version. (The down-migration is the same in both versions.)
```

#### #42 - 01/27/2015 06:28 PM - Radhika Chippada

Tom, addressed all your comments. I also simplified has_description check further. Thanks.

#### #43 - 01/29/2015 03:52 PM - Tom Clegg

Looking at 4523-full-text-search at 316eca14...

Currently, the filter attribute(s) provided by the client are ignored, effectively coerced to "any". I think it's better if filters=[['any','@@','foo']] searches all fulltext-searchable columns, but filters=[['name','@@','foo']] returns an error (until/unless we support that kind of filtering).

Rather than introducing an extra condition around the per-column loop, you could just feed it an empty array, something like this:

```
cond_out = [] # (moved this up so we can append stuff to it earlier)
if operator == '@@'
  if attrs_in != 'any'
    raise ArgumentError.new("Full text search on individual columns is not supported")
  end
  attrs = [] # this skips the generic per-column operator loop below
  cond_out << ... # do the full-text search stuff here
elsif attrs_in == 'any'
  ...
```

I haven't scrutinized the test cases but I noticed some stuff like assert_equal true, results.length>0, ... which seems like it should be assert_not_empty results, ... (and similar for assert_equal 0, results.length)...

#### #44 - 01/29/2015 06:17 PM - Radhika Chippada

Tom,

- Added the ArgumentError when individual columns are used in filter
- Added tests to cover this condition
- Updated test to use assert_empty, refute_empty, assert_includes, assert_not_includes as applicable. Yes, this is much better. Thanks.

#### #45 - 02/10/2015 03:46 PM - Tom Clegg

LGTM @ 330a46e