How do I do full-text searching in Ruby on Rails?

Asked 16 years, 3 months ago Modified 4 years, 5 months ago

Viewed 11k times

I would like to do full-text searching of data in my Ruby on Rails application. What options exist?

12

ruby-on-rails full-text-search

Share

Improve this question

Follow

asked Sep 6, 2008 at 16:51

sock

1,054 ● 2 ● 8 ● 17

8 Answers

Sorted by: Highest score (default)

20

There are several options available and each have different strengths and weaknesses. If you would like to add full-text searching, it would be prudent to investigate each a little bit and try them out to see how well it works for you in your environment.

MySQL has built-in support for full-text searching. It has online support meaning that when new records are added to the database, they are automatically indexed and will

be available in the search results. The documentation has more details.

`acts_as_tsearch` offers a wrapper for similar built-in functionality for recent versions of PostgreSQL

For other databases you will have to use other software.

Lucene is a popular search provider written in Java. You can use Lucene through its search server Solr with Rails using `acts_as_solr`.

If you don't want to use Java, there is a port of Lucene to Ruby called Ferret. Support for Rails is added using the `acts_as_ferret` plugin.

Xapian is another good option and is supported in Rails using the `acts_as_xapian` plugin.

Finally, my preferred choice is Sphinx using the Ultrasphinx plugin. It is extremely fast and has many options on how to index and search your databases, but is no longer being actively maintained.

Another plugin for Sphinx is Thinking Sphinx which has a lot of positive feedback. It is a little easier to get started using Thinking Sphinx than Ultrasphinx. I would suggest investigating both plugins to determine which fits better with your project.

Share Improve this answer

Follow

edited Apr 22, 2015 at 22:30

community wiki
6 revs
sock

8

I can recommend Sphinx. Ryan Bates has a great screencast on using the Thinking Sphinx plugin to create a full-text search solution.

Share Improve this answer

Follow

answered Sep 6, 2008 at 17:04

John Topley
115k • 47 • 199 • 240

5

You can use Ferret (which is Lucene written in Ruby). It integrates seamless with Rails using the acts_as_ferret mixin. Take a look at "How to Integrate Ferret With Rails". A alternative is Sphinx.

Share Improve this answer

Follow

edited Feb 2, 2014 at 20:13

BenMorel
36.4k • 51 • 202 • 334

answered Sep 6, 2008 at 16:55

marcospereira
12.2k • 3 • 48 • 53

ferret was a great gem but unfortunately doesn't work any longer since many years now – peter Jun 20, 2021 at 20:23

Hey @peter, you are right. This answer was written in 2008, though. :-) – marcospereira Jun 21, 2021 at 18:57

my comment was to prevent people from trying it at this date cause it won't work, I have been on the lookout for a replacement since long, just yesterday I found the gem picky that seems to do the job – peter Jun 22, 2021 at 15:59

3

Two main options, depending on what you're after.

1) Full Text Indexing and `MATCH() AGAINST()`.

If you're just looking to do a fast search against a few text columns in your table, you can simply use a full text index of those columns and use `MATCH() AGAINST()` in your queries.

1. Create the full text index in a migration file:

```
add_index :table, :column, type: :fulltext
```

2. Query using that index:

```
where( "MATCH( column ) AGAINST( ? )", term )
```

2) ElasticSearch and Searchkick

If you're looking for a full blown search indexing solution that allows you to search for any column in any of your records while still being lightning quick, take a look at ElasticSearch and Searchkick.

ElasticSearch is the indexing and search engine.

Searchkick is the integration library with Rails that makes it very easy to index your records and search them.

Searchkick's README does a fantastic job at explaining how to get up and running and to fine tune your setup, but here is a little snippet:

1. Install and start ElasticSearch.

```
brew install elasticsearch
brew services start elasticsearch
```

2. Add `searchkick` gem to your bundle:

```
bundle add searchkick --strict
```

   The `--strict` option just tells Bundler to use an exact version in your Gemfile, which I highly recommend.

3. Add `searchkick` to a model you want to index:

```
class MyModel < ApplicationRecord
  searchkick
end
```

4. Index your records.

```
MyModel.reindex
```

5. Search your index.

```
matching_records = MyModel.search( "term" )
```

Share Improve this answer Follow

edited Jul 6, 2020 at 20:50

answered Dec 29, 2019 at 21:07

Joshua Pinter

47.3k ● 23 ● 255 ● 252

I've been compiling a [list of the various Ruby on Rails search options in this other question](). I'm not sure how, or if to combine our questions.

1

Share Improve this answer

Follow

edited May 23, 2017 at 11:59

Community Bot

1 ● 1

answered Sep 16, 2008 at 18:33

Otto

19.3k ● 16 ● 58 ● 62

That is a more comprehensive list, but lacks some details on the strengths and weaknesses of each; this question as well. What I would like to see is a list of each plugin with information about what are its strength and weaknesses, and links to documentation and tutorials. – sock Sep 17, 2008 at 16:52

It depends on what database you are using. I would recommend using Solr as it offers up a lot of nice options. The downside is you have to run a separate process for it. I have used Ferret as well, but found it to be less stable in terms of multi-threaded access to the index. I haven't tried Sphinx because it only works with MySQL and Postgres.

1

Share Improve this answer

Follow

answered Sep 23, 2008 at 14:21

MattMcKnight

8,260 ● 31 ● 35

1

Just a note for future reference: Ultra Sphinx is no longer being maintained. Thinking sphinx is its replacement. Although it lacks several features at this time like excerpting which Ultra sphinx had, it makes up for it in other features.

Share Improve this answer

Follow

answered Mar 11, 2009 at 13:04

iros

1,045 ● 1 ● 8 ● 8

1

I would recommend acts_as_ferret as I am using it for Scrumpad project at work. The indexing can be done as a separate process which ensures that while re-indexing we can still use our application. This can reduce the downtime of website. Also the searching is much faster. You can search through multiple model at a time and have your results sorted out by the fields you prefer.

Share Improve this answer

Follow

answered Aug 29, 2009 at 17:27

user116218

29 ● 2