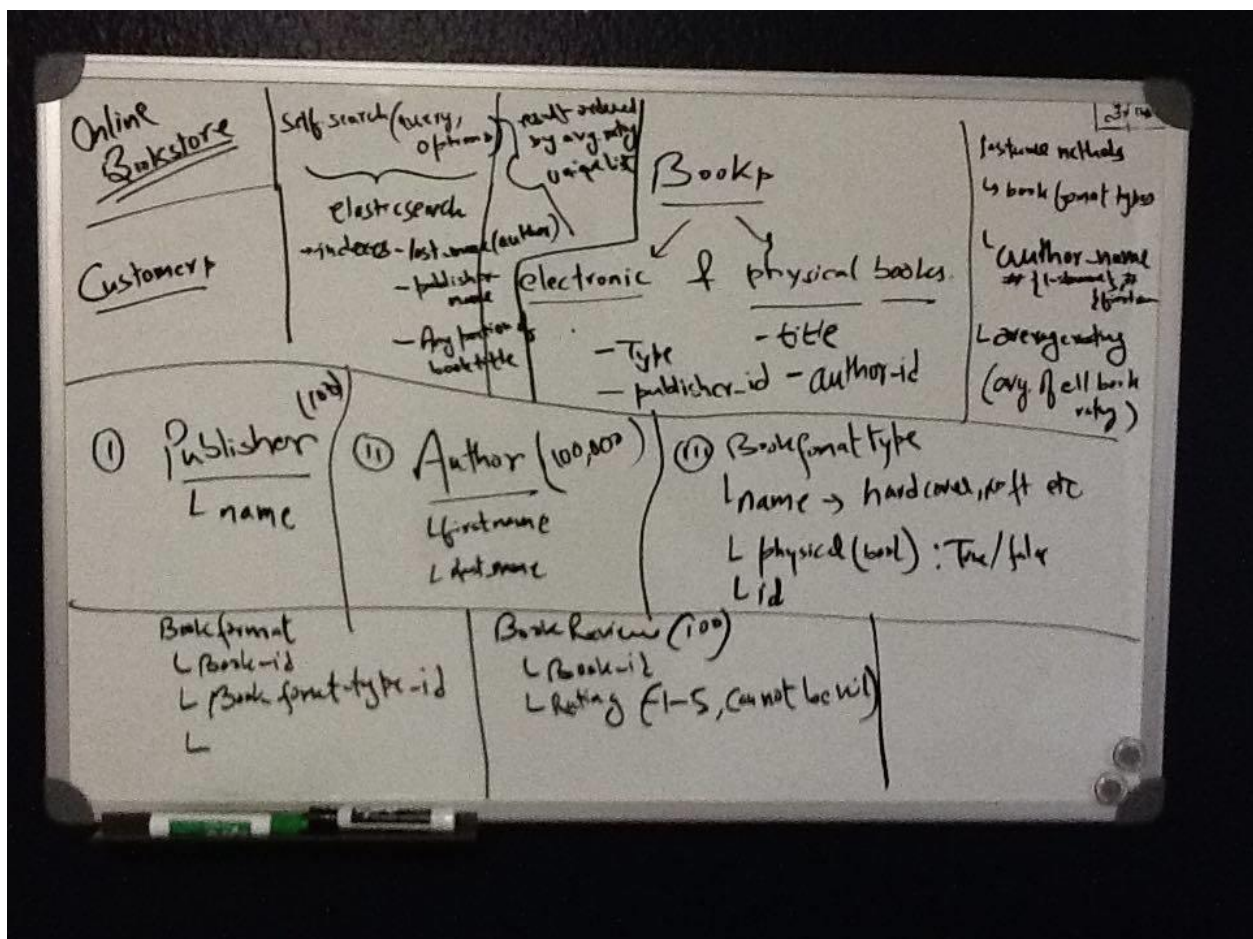


1. The code is mainly in app/models folder.
2. I used TDD strategy for development, tests are in spec folder.
3. Run the tests through: **bundle exec rspec**
4. I am using factories instead of fixtures
5. To test elasticsearch in Rspec, first run an instance of elastic search at localhost:9200

## STEPS

1. Starting with the project, first of all I read the requirements twice and tried to list out the entities and their required behaviors on my whiteboard in a rough fashion:



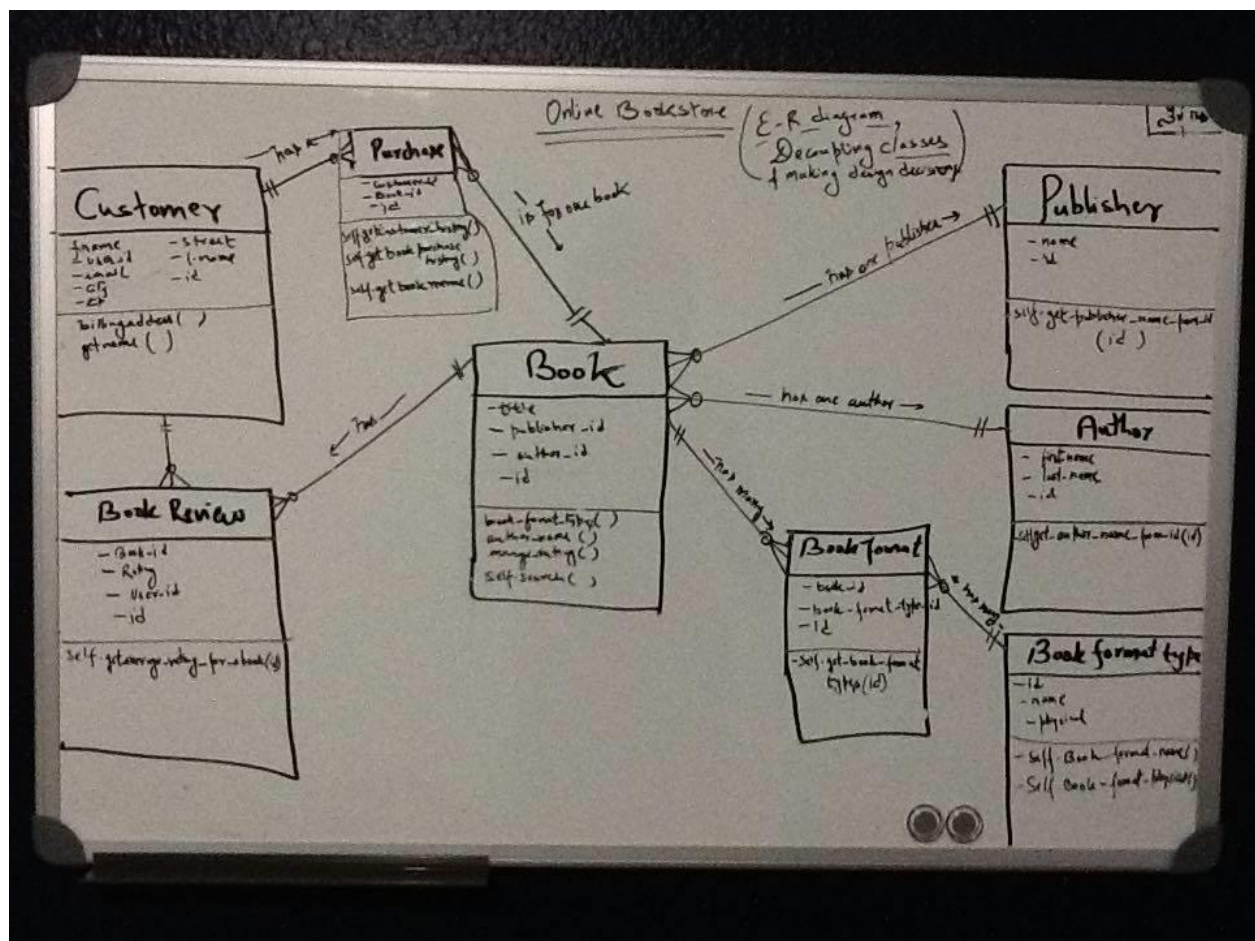
It's mentioned that other entities have already been designed and I need to add only Book model but I felt that in order to make the application good, make the code base clean, entities loosely coupled,

showcase my design skills and follow the object oriented design principles efficiently, I may have to alter the design decisions in other classes and so should start from blank project and implement well tested interfaces for classes but concentrate more on the requirements.

1. Through an entity relationship diagram, I decided to understand the entity relationships and add few more entities which make the application more real. Example: purchase, customers etc.
2. Early in my application design process, with the goal of achieving loose coupling, I tried to add few methods where it should actually belong and decided to use interfaces for inter class communications. The methods added in other entities in the image below represents that thought process.

For the sake of completeness and helping me out with testing, I created other entities minimally in my app.

Later on, one can decide if we want to keep them in the final app or follow a micro-service architecture design where I believe my team is only handling the Book model and other teams are handling other models and have already optimized it and are allowing the communications needed for the Book to function through well designed public interfaces.



### 3. I generated a new rails 5 project using:

*Rails g terakeet\_bookstore*

### 4. I generated the models

*rails g model Publisher name:string*

*rails g model Customer first\_name:string last\_name:string email:string*

*rails g model Author first\_name:string last\_name:string*

*rails g model Purchase customer:references book:references*

*rails g model BookReviews rating:integer customers:references book:references*

*rails g model BookFormatType name:string physical:boolean*

*rails g model BookFormat book:references bookformattype:references*

Finally, the Book model:

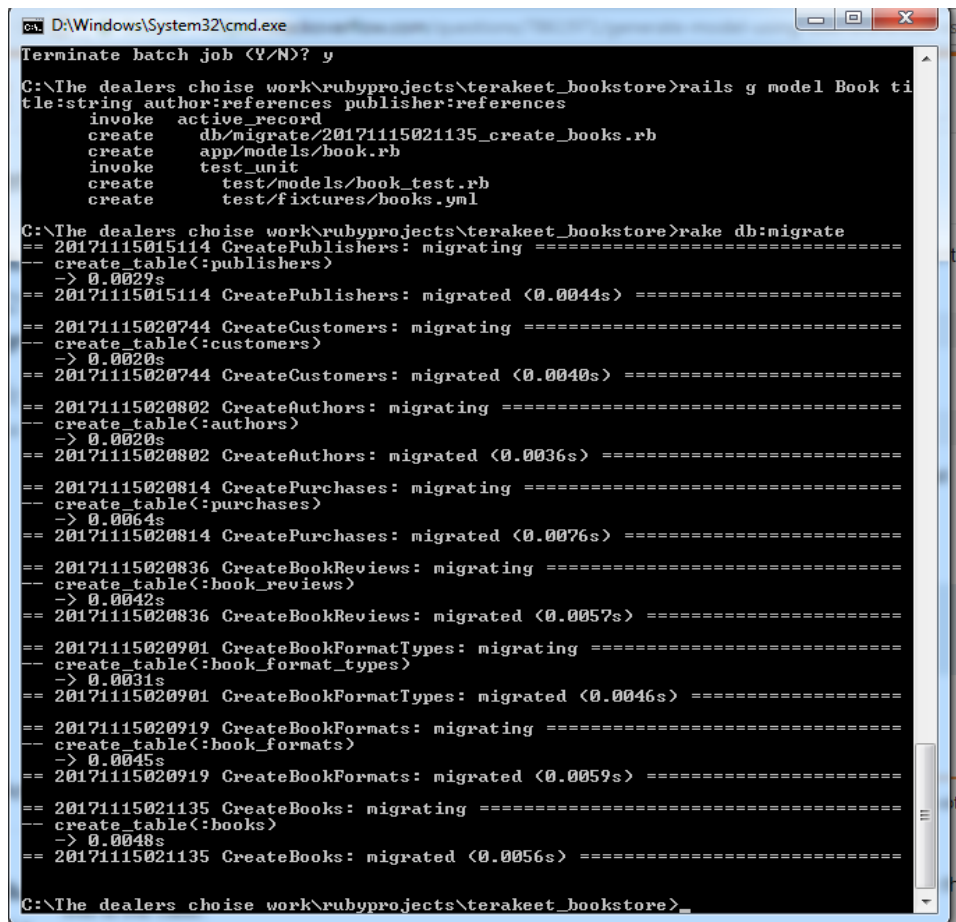
*rails g model Book title:string author:references publisher:references*

*Note: I had to add one more migration to change column name because of an error I received while establishing relationships among classes and testing through Rspec.*

```
class FixcolumnName < ActiveRecord::Migration[5.1]
  def change
    rename_column :book_formats, :bookformattype_id, :book_format_type_id
  end
end
```

### 5. After the migrations were ready I ran *rake db:migrate*

6. I decided to use **through in models instead of has\_many\_and\_belongs\_to relationships** when



```
D:\Windows\System32\cmd.exe
Terminate batch job (Y/N)? y
C:\The dealers choice work\rubyprojects\terakeet_bookstore>rails g model Book title:string author:references publisher:references
  invoke  active_record
  create  db/migrate/20171115021135_create_books.rb
  create  app/models/book.rb
  invoke  test_unit
  create  test/models/book_test.rb
  create  test/fixtures/books.yml

C:\The dealers choice work\rubyprojects\terakeet_bookstore>rake db:migrate
== 20171115015114 CreatePublishers: migrating =====
-- create_table(:publishers)
--> 0.0029s
== 20171115015114 CreatePublishers: migrated <0.0044s> =====
== 20171115020744 CreateCustomers: migrating =====
-- create_table(:customers)
--> 0.0020s
== 20171115020744 CreateCustomers: migrated <0.0040s> =====
== 20171115020802 CreateAuthors: migrating =====
-- create_table(:authors)
--> 0.0020s
== 20171115020802 CreateAuthors: migrated <0.0036s> =====
== 20171115020814 CreatePurchases: migrating =====
-- create_table(:purchases)
--> 0.0064s
== 20171115020814 CreatePurchases: migrated <0.0076s> =====
== 20171115020836 CreateBookReviews: migrating =====
-- create_table(:book_reviews)
--> 0.0042s
== 20171115020836 CreateBookReviews: migrated <0.0057s> =====
== 20171115020901 CreateBookFormatTypes: migrating =====
-- create_table(:book_format_types)
--> 0.0031s
== 20171115020901 CreateBookFormatTypes: migrated <0.0046s> =====
== 20171115020919 CreateBookFormats: migrating =====
-- create_table(:book_formats)
--> 0.0045s
== 20171115020919 CreateBookFormats: migrated <0.0059s> =====
== 20171115021135 CreateBooks: migrating =====
-- create_table(:books)
--> 0.0048s
== 20171115021135 CreateBooks: migrated <0.0056s> =====

C:\The dealers choice work\rubyprojects\terakeet_bookstore>
```

**needed for many to many mappings.**

7. I also added **validations** for fields I thought were important in the models. All models are under **app/models** folder.
8. Now that I had all the models generated, instead of generating a full-fledged application with controllers and view, I decided to go through test driven development to limit my efforts to **model methods only**.

I decided to use:

- **gem 'rspec-rails', '~> 3.6'**
- **gem "factory\_bot\_rails""~> 4.0"**
- **gem 'database\_cleaner'**

gems to have more control and loose coupling over my testing( instead of using other testing frameworks or fixtures). The setting and configurations are specified in spec and rspec helpers.

**Note: I followed a TDD strategy. All the tests and factories are in spec folder.**

9. Starting from more independent models like authors and publishers I used TDD for generating the methods and their models needed.
10. I tried to minimize dependencies and aid in future ease of changing codes and used object oriented design, refactoring, small functions, broken down functions depending upon the task levels and followed single responsibility principles whenever needed.
11. I used private methods to act as interfaces between classes. These are more susceptible to changes. I only provided concrete methods as public interface for each class and let them pass message only through interface.

```
C:\The dealers chose work\rubyprojects\terakeet_bookstore>
C:\The dealers chose work\rubyprojects\terakeet_bookstore>bundle exec rspec
.....
Finished in 43.11 seconds (files took 7.73 seconds to load)
25 examples, 0 failures

C:\The dealers chose work\rubyprojects\terakeet_bookstore>bundle exec rspec
.....F.....
Failures:
  1) Book search should search for books for given query and given title only optionsql_s
     Failure/Error: expect(Book.sql_search('Secrets', title_only: false, book_format_type_id: book_format_type1.id)).to match_array([book1, book4, book5])
     expected collection contained: [#<Book id: 2, title: "The Secrets You Keep: A Novel", author_id: 3, publisher_id: 3, created_at: "2017-11-20 21:44:38", updated_at: "2017-11-20 21:44:38">]
     actual collection contained: [#<Book id: 2, title: "The Secrets You Keep: A Novel", author_id: 3, publisher_id: 3, created_at: "2017-11-20 21:44:38", updated_at: "2017-11-20 21:44:38">, #<Book id: 7, title: "Fictional Book 1", author_id: 7, publisher_id: 7, created_at: "2017-11-20 21:44:38", updated_at: "2017-11-20 21:44:38">]
     the missing elements were: [#<Book id: 5, title: "Fictional Book 1", author_id: 9, publisher_id: 9, created_at: "2017-11-20 21:44:38", updated_at: "2017-11-20 21:44:38">]
     # ./spec/book_spec.rb:324:in 'block <3 levels> in <top (required)>'

Finished in 35.31 seconds (files took 9.55 seconds to load)
25 examples, 1 failure

Failed examples:
rspec ./spec/book_spec.rb:220 # Book search should search for books for given query and given title only optionsql_s

C:\The dealers chose work\rubyprojects\terakeet_bookstore>bundle exec rspec
.....
Finished in 35.63 seconds (files took 10.14 seconds to load)
25 examples, 0 failures

C:\The dealers chose work\rubyprojects\terakeet_bookstore>bundle exec rspec
.....
Finished in 30.64 seconds (files took 7.77 seconds to load)
25 examples, 0 failures

C:\The dealers chose work\rubyprojects\terakeet_bookstore>
```



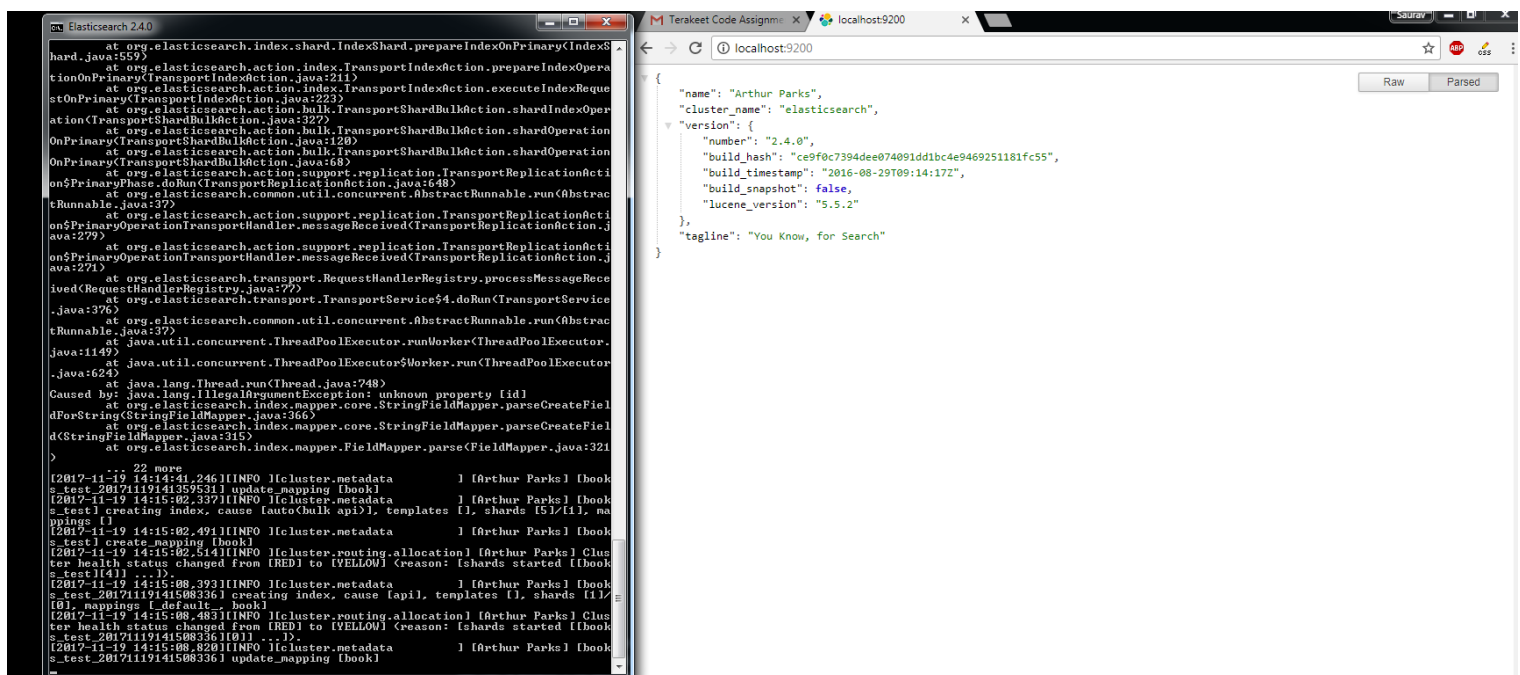
## Elastic Search:

Note: Although it was asked to generate one class method `search()`. I decided to provide two search methods in my public interface instead.

- One is simple `sql_search()` class methods which implements SQL based searching. And other is `search_instead_with_elastic_search()` which is better performing with respect to bigger data and faster text based search.
- What should we use will depend upon results from performance testing.
- My reason is that it will provide the team more flexibility in choosing the right search method to use based upon the use case, production server data size, memory available, memory-performance analysis, use of caching strategies etc.

Also, since the number of rows in each tables in the production app was specified in the requirement document and elastic search indexing takes a lot of memory, I used elastic search on authors (100000) and books (100000, given each author has at least one book) and publishers when I needed to do a text based search. Otherwise, I relied on SQL based searching to filter on to the results of elastic search.

Note: For elastic search, I have an instance running in my local machine as shown in the figure below:



The screenshot shows two windows. The left window is the Elasticsearch 2.4.0 console, displaying a stack trace for a `java.lang.IllegalArgumentException: unknown property [id]` error. The right window is a web browser at `localhost:9200` showing a JSON search result for 'Arthur Parks'.

```
at org.elasticsearch.index.shard.IndexShard.prepareIndexOnPrimary<IndexShard.java:559>
at org.elasticsearch.action.index.TransportIndexAction.prepareIndexOperationOnPrimary<TransportIndexAction.java:211>
at org.elasticsearch.action.index.TransportIndexAction.executeIndexRequestOnPrimary<TransportIndexAction.java:223>
at org.elasticsearch.action.bulk.TransportShardBulkAction.shardIndexOperationOnPrimary<TransportShardBulkAction.java:327>
at org.elasticsearch.action.bulk.TransportShardBulkAction.shardOperationOnPrimary<TransportShardBulkAction.java:120>
at org.elasticsearch.action.bulk.TransportShardBulkAction.shardOperationOnPrimary<TransportShardBulkAction.java:56>
at org.elasticsearch.action.support.replication.TransportReplicationAction$PrimaryPhase.doRun<TransportReplicationAction.java:648>
at org.elasticsearch.common.util.concurrent.AbstractRunnable.run<AbstractRunnable.java:37>
at org.elasticsearch.action.support.replication.TransportReplicationAction$PrimaryOperationTransportHandler.messageReceived<TransportReplicationAction.java:279>
at org.elasticsearch.action.support.replication.TransportReplicationAction$PrimaryOperationTransportHandler.messageReceived<TransportReplicationAction.java:271>
at org.elasticsearch.transport.RequestHandlerRegistry.processMessageReceived<RequestHandlerRegistry.java:72>
at org.elasticsearch.transport.TransportService$4.doRun<TransportService.java:376>
at org.elasticsearch.common.util.concurrent.AbstractRunnable.run<AbstractRunnable.java:37>
at java.util.concurrent.ThreadPoolExecutor.runWorker<ThreadPoolExecutor.java:1149>
at java.util.concurrent.ThreadPoolExecutor$Worker.run<ThreadPoolExecutor.java:624>
at java.lang.Thread.run<Thread.java:748>
Caused by: java.lang.IllegalArgumentException: unknown property [id]
at org.elasticsearch.index.mapper.core.StringFieldMapper.parseCreateFieldForString<StringFieldMapper.java:366>
at org.elasticsearch.index.mapper.core.StringFieldMapper.parseCreateField<StringFieldMapper.java:315>
at org.elasticsearch.index.mapper.FieldMapper.parse<FieldMapper.java:321>
... 22 more
[2017-11-19 14:14:41.246][INFO][cluster.metadata] [Arthur Parks] book
s_test_20171119141359531 update_mapping [book]
[2017-11-19 14:15:02.337][INFO][cluster.metadata] [Arthur Parks] book
s_test1 creating index, cause [auto(bulk api)], templates [], shards [5]/[1], mappings []
[2017-11-19 14:15:02.491][INFO][cluster.metadata] [Arthur Parks] book
s_test1 create_mapping [book]
[2017-11-19 14:15:02.514][INFO][cluster.routing.allocation] [Arthur Parks] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [book
s_test1[4]] ...]).
[2017-11-19 14:15:08.393][INFO][cluster.metadata] [Arthur Parks] book
s_test_201711191415083361 creating index, cause [api], templates [], shards [1]/[0], mappings [default_ book]
[2017-11-19 14:15:08.483][INFO][cluster.routing.allocation] [Arthur Parks] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [book
s_test_201711191415083361[0]] ...]).
[2017-11-19 14:15:08.820][INFO][cluster.metadata] [Arthur Parks] book
s_test_201711191415083361 update_mapping [book]
```

```
{
  "name": "Arthur Parks",
  "cluster_name": "elasticsearch",
  "version": {
    "number": "2.4.0",
    "build_hash": "ce9f0c7394dee074091dd1bc4e9469251181fc55",
    "build_timestamp": "2016-08-29T09:14:17Z",
    "build_snapshot": false,
    "lucene_version": "5.5.2"
  },
  "tagline": "You Know, for Search"
}
```

**Limitations: What I couldn't do but if given more time I will be able to finish:**

For sql exact matching works great but I tried to work with exact matching for publisher and author last name with elastic search gem Searchkick and tried these options as suggested by [Searchkick](#) documentation but for now exact matching isn't working. I am working on it.

I have raised the issue on the github and asked for help. If given more time, I will be able to find the right solution which pass my tests.

*I know in pure Elasticsearch REST interface, using term matching is the way.*

**These queries don't work.**

```
self.search(query, fields: [{title: :word_middle}, {publisher_name: :word}, {last_name: :word}]).records
```

```
self.search(query, fields: [:title, {publisher_name: :exact}, {last_name: :exact}]).records
```

```
self.search(query, where: [{last_name: /\b"#{query}"\b/},{publisher_name: /\b"#{query}"\b/},{title: /\b"#{query}"\b/}]).records
```

---

All code have been extensively tested using Rspec (Please see spec folder) and then refactored.

If I missed something or did something wrong please let me know. I know I am capable of producing great results.

I really want to work for Terakeet and I will do what it takes.

Also, I miss the snow in Syracuse and want to be back there and work there, grow, learn and contribute with your team and I feel I deserve it. I want the opportunity.

Thank you.

Yours truly,

Saurav Prakash

Blog: <http://www.onceaday.today/>