

JRedisearch Workshop

By David Parry



Principal Software Engineer at Mutualink Inc.

Worked for a range of companies from startups to big four consulting firms.

Contributes to JRedisearch and Jedis

<http://www.davidparry.com> <- more here



Redis Overview

- Data Structures Server.
- Commands to access mutable data structures.
- Prefers to store all data in memory faster.
- Data structures stress on memory efficiency, smaller footprint.
- Features typical database; replication, tunable durability, cluster, high availability.
- CRDB – Conflict Free Replicated Database (Paid Version)

Redisearch Module Overview

- Developed and maintained by Redis Labs
- Redis Powered Search Engine
 - Open-Source Full-Text and Secondary Index engine
- Does not use internal data structures like sorted sets
 - Exact phrase matching and numeric filtering for text queries, that are not possible or efficient with traditional Redis search approaches example for a key `'(hello world)|(hola mundo)'`
- Reference Material: <https://oss.redislabs.com/redisearch>

Main Features

Full-Text indexing of multiple fields in documents

Incremental indexing without performance loss

Document ranking (manually at index time).

Optional query clauses

Complex boolean queries with AND, OR, NOT operators between sub-queries

Auto-complete suggestions (with fuzzy prefix suggestions)

Prefix based searches

Field weights

Exact Phrase Search, Slop based search.

Numeric filters and ranges

Support for custom functions for query expansion and scoring (see Extensions)

Stemming based query expansion in many languages (using Snowball)

Geo filtering using Redis' own Geo-commands

Unicode support (UTF-8 input required)

Retrieve full document content or just ids

Partial and conditional document updates

Document deletion and updating with index garbage collection

Optional query clauses

JRedisearch Overview

- Java library abstracting the API of the RediSearch Redis module
- <https://github.com/RedisLabs/JRediSearch>
- **Topics to Cover**
 - Getting Started
 - Client Usage
 - Schema
 - Index Usage
 - Documents
 - Query Usage
 - Suggestion Usage

Final Result

← → ↻ ⓘ localhost:8080 ☆



David Parry

Search 'The Art of Alchimy Book'

Search

Score	Chapter	Line	ID	Text
9.5	1	6	1:6	ThE Philosopher saith. It is true, to wit, that the Arte of
8	29	1533	29:1533	proper principles and conditions of Arts and Sciences they disagree,
3	29	1387	29:1387	These things are almost as much as nature or Art are able to performe.
3	29	1388	29:1388	But yet the last decree, wherein the perfection of Art can doo oughts
3	1	8	1:8	detestation of them that affirme this Art to bee lying, that is, false.
3	14	398	14:398	which are orderly dissolued and congealed, wherein neither addition
3	14	396	14:396	a greater quantity partaker after these two. I vnderstand by these
3	14	390	14:390	vtttered or named. We say then that the great work contained in it
3	14	394	14:394	betweene themselves, and so likewise are the other two. And either
3	14	395	14:395	of these double quãtities hath another quantity partaker, which is

Getting Started

➤ Steps to readme:

➤ Git install - <https://git-scm.com/downloads>

➤ Verify – `$ git clone git@github.com:davidparry/JRediSearchWorkshop.git`

➤ Follow the README.md for further instructions

Client Usage



Goal: Connect to Redisearch and verify connectivity

Reference Material:

https://oss.redislabs.com/redisearch/java_client/

<http://davidparry.com/storage/jredisearch-javadoc-v0-19-0/docs/io/redisearch/Client.html>

Index Usage



Goal: Demonstrate a better approach to check connectivity

See post @ <http://davidparry.com/blog/2018/12/2/testing-conductivity-in-jredissearch-to-redis-without-a-prev.html>

Schema

Goal: Define and create a Schema in Redisearch

Reference Material: <https://oss.redislabs.com/redisearch/Commands/#ftcreate>

<https://oss.redislabs.com/redisearch/Commands/#ftinfo>

<http://davidparry.com/storage/jredisearch-javadoc-v0-19-0/docs/io/redisearch/client/Client.html#createIndex-io.redisearch.Schema-io.redisearch.client.Client.IndexOptions->

Documents

Goal: To add a simple Document to the Index and then a more complex with a Payload

Reference Material:

<https://oss.redislabs.com/rediseach/Commands/#ftadd>

<https://oss.redislabs.com/rediseach/payloads/>

<http://davidparry.com/storage/jrediseach-javadoc-v0-19-0/docs/io/rediseach/Client.html#addDocument-io.rediseach.Document-io.rediseach.client.AddOptions->

<http://davidparry.com/storage/jrediseach-javadoc-v0-19-0/docs/io/rediseach/Document.html#Document-java.lang.String-java.util.Map-double-byte:A->

Query Usage



Goal: retrieve a document using a simple term using a query object

Reference Material:

https://oss.redislabs.com/rediseach/Query_Syntax/

<http://davidparry.com/storage/jrediseach-javadoc-v0-19-0/docs/io/rediseach/Client.html#search-io.rediseach.Query->

Suggestion Usage

Goal: add suggestions to the index and retrieve suggested words that are fuzzy

Note: *Adds a suggestion string to an auto-complete suggestion dictionary. This is disconnected from the index definitions, and leaves creating and updating suggestions dictionaries to the user*

Reference Material:

<https://oss.redislabs.com/rediseach/Commands/#ftsugadd>

<http://davidparry.com/storage/jrediseach-javadoc-v0-19-0/docs/io/rediseach/Client.html#addSuggestion-io.rediseach.Suggestion-boolean->

<http://davidparry.com/storage/jrediseach-javadoc-v0-19-0/docs/io/rediseach/Client.html#getSuggestion-java.lang.String-io.rediseach.client.SuggestionOptions->

Recap

- Getting Started
- Client Usage
- Schema
- Index Usage
- Documents
- Query Usage
- Suggestion Usage

Q&A

