Jurnal Teknik Informatika (JUTIF)
Vol. 4, No. 5, October 2023, pp. 1081-1087

DOI: https://doi.org/10.52436/1.jutif.2023.4.5.959
p-ISSN: 2723-3863
e-ISSN: 2723-3871

IMPLEMENTATION OF TEXT INDEXING SYSTEM IN WEB-BASED DOCUMENT SEARCH APPLICATION USING MONGODB

Frankie[*1], Yeremia Alfa Susetyo[2]

[1,2]Informatics Engineering, Faculty of Information Technology, Universitas Kristen Satya Wacana, Indonesia
Email: [1]672019165@student.uksw.edu, [2]yeremia.alfa@uksw.edu

(Article received: March 18, 2023; Revision: March 27, 2023; published: October 15, 2023)

Abstract

The rapid growth of information technology has led to an increase in the amount of data stored in databases every day. Relational databases (SQL) that have been in use for a long time are now being developed with the emergence of NoSQL databases such as MongoDB. MongoDB stores data in BSON format and has a Text Indexes feature that is useful for speeding up text search on string content. This feature is particularly useful in searching for data in the form of texts or strings in large quantities. MongoDB's Text Indexes have a flexible schema that does not require a strict schema structure to index text data, unlike SQL databases that require columns with the appropriate data type to perform indexing. MongoDB's Text Indexes support more languages than SQL because they use an open-source text search engine called Apache Lucene. In this study, the researcher will implement Text Indexing on document data (PDF) that has been converted into text, then inserted into MongoDB before indexing. Afterward, the researcher will compare the performance of search queries between indexed and non-indexed data in MongoDB in terms of speed. The comparison results will be presented in tables and graphs to facilitate understanding. Based on the research conducted, it can be concluded that the use of the text indexing feature in MongoDB can speed up keyword or string search time. In the experiment conducted using 5000 data records, the results showed that the use of text indexing for searching 1 keyword resulted in a search speed improvement of 11705,88%, for searching 2 keywords it was 60833,33%, and for searching 3 keywords it was 44320%.

Keywords: *Data Search, Fulltext Search, MongoDB, Python, Text Indexing.*

IMPLEMENTASI SISTEM *TEXT INDEXING* PADA APLIKASI PENCARIAN DATA DOKUMEN MENGGUNAKAN MONGODB BERBASIS WEB

Abstrak

Pertumbuhan teknologi informasi yang sangat pesat membuat jumlah data yang tersimpan dalam *database* semakin banyak setiap harinya. Basis data relasional (SQL) yang sudah lama digunakan kini mengalami perkembangan dengan munculnya *database* NoSQL seperti MongoDB. MongoDB menyimpan data dalam format BSON dan memiliki fitur *Text Indexes* yang berguna untuk mempercepat pencarian teks pada konten *string*. Fitur ini sangat berguna dalam mencari data dalam bentuk teks atau *string* dalam jumlah yang besar. Text Indexes MongoDB memiliki skema yang fleksibel sehingga tidak memerlukan struktur skema yang ketat untuk mengindeks data teks dibandingkan dengan database SQL yang memerlukan kolom dengan tipe data yang tepat untuk melakukan indeks. Text Indexes MongoDB mendukung bahasa yang lebih banyak dibandingkan SQL karena menggunakan mesin pencarian teks *open source* yang disebut Apache Lucene. Dalam penelitian ini, peneliti akan mengimplementasikan *Text Indexing* pada data dokumen (PDF) yang sudah dikonversi menjadi teks, kemudian dimasukkan ke dalam MongoDB sebelum melakukan *indexing*. Setelah itu, peneliti akan melakukan perbandingan performa *query* pencarian antara data yang sudah terindeks dan belum terindeks dalam *database* MongoDB dari segi kecepatan. Hasil perbandingan akan dibuat dalam bentuk tabel dan grafik agar lebih mudah dipahami. Berdasarkan hasil penelitian yang dilakukan, dapat disimpulkan bahwa penggunaan fitur *text indexing* pada MongoDB dapat mempercepat waktu pencarian kata atau *string*. Dalam percobaan yang dilakukan dengan menggunakan 5000 *record* data, didapatkan hasil bahwa penggunaan *text indexing* pada pencarian 1 *keyword* menghasilkan peningkatan kecepatan pencarian sebesar 11705,88%, pada pencarian 2 *keyword* sebesar 60833,33%, dan pada pencarian 3 *keyword* sebesar 44320%.

Kata kunci: *Fulltext Search, MongoDB, Pencarian Data, Python, Text Indexing.*

1. PENDAHULUAN

1.1. Latar Belakang

Pesatnya pertumbuhan teknologi informasi menghasilkan miliaran data setiap harinya [1]. Sehingga seiring berjalannya waktu, data yang masuk ke dalam *database* mengalami peningkatan besar. Peningkatan data yang besar dapat mempengaruhi kecepatan pengaksesan *database*.

Basis data relasional (SQL) telah menjadi andalan pemrosesan data selama beberapa dekade. Di sisi lain, terjadi perkembangan dalam pengelolaan basis data seperti NoSQL (*Non SQL*) dengan menggunakan metode baru yaitu *key-value*, *document-oriented*, dan *graph* [2].

Hingga saat ini, masih banyak yang menggunakan basis data dengan model relasional untuk pengolahan data dalam jumlah besar [3]. Permasalahan yang rentan terjadi adalah penyimpanan data dengan jumlah besar dapat menjadi kendala saat melakukan pencarian data [4]. Semakin banyak data, semakin lama waktu yang dibutuhkan pengguna untuk menemukan data.

Salah satu *database* NoSQL yang sangat populer adalah MongoDB. Berbeda dengan *database* SQL yang menyimpan data dengan relasi tabel, MongoDB menyimpan data dalam format BSON (Binary JSON). Pada MongoDB, terdapat fitur *Text Indexes* untuk mendukung fungsi pencarian teks pada konten *string*. *Text Indexes* dapat mempercepat pencarian data dan dapat digunakan tanpa menggunakan *dependency* pihak ketiga. Fitur *Text Indexes* MongoDB dapat digunakan dalam pencarian data dalam bentuk teks / *string* dalam jumlah yang besar. Text Indexes MongoDB memiliki skema yang fleksibel sehingga tidak memerlukan struktur skema yang ketat untuk mengindeks data teks dibandingkan dengan database SQL yang memerlukan kolom dengan tipe data yang tepat untuk melakukan indeks. Text Indexes MongoDB mendukung bahasa yang lebih banyak dibandingkan SQL karena menggunakan mesin pencarian teks *open source* yang disebut Apache Lucene [5][6].

Dengan adanya fitur *Text Indexes* dari MongoDB, peneliti akan mengimplementasikan *Text Indexing* ke dalam data dokumen (PDF) yang sudah dikonversi menjadi teks. Teks tersebut dimasukkan kedalam *database* MongoDB sebelum dilakukan *indexing*. Setelah itu, dilakukan perbandingan performa *query* pencarian antara data yang sudah terindeks ataupun belum terindeks. Perbandingan dilakukan dari segi kecepatan pada *database* MongoDB.

1.2. Kajian Pustaka

Python merupakan bahasa pemrograman tingkat tinggi yang sangat populer. Python memiliki beberapa keunggulan yaitu dapat digunakan dalam pembangunan *website*, perangkat lunak, perangkat keras, dan juga *video game*. Python juga memiliki struktur kode yang mudah dipahami dan memiliki *library* yang sangat banyak [7].

Flask merupakan *micro framework* yang digunakan untuk membangun aplikasi web dengan bahasa pemograman Python. Flask menggunakan 2 *library* eksternal yaitu WSGI toolkit dan Jinja2 [8]. Flask merupakan *micro framework* karena tidak mengharuskan pengguna dalam menggunakan alat atau pustaka tertentu. Flask digunakan untuk membuat kerangka dan mengatur tampilan dalam sebuah aplikasi. Flask sangat fleksibel dan terukur sehingga memungkinkan pengguna untuk menggunakan ekstensi dan dependensi pihak ketiga yang sudah disediakan oleh Flask [9].

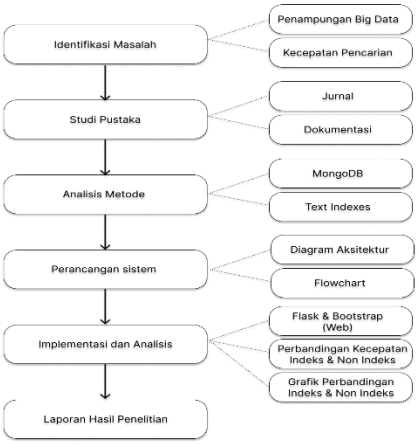
MongoDB adalah *database* NoSQL berbasis *document based*. NoSQL adalah singkatan dari *Not Only* SQL, artinya sistem *database* tidak harus menggunakan perintah SQL saja untuk melakukan proses manipulasi data. MongoDB berbasis *document based* berarti MongoDB tidak memiliki tabel, kolom, ataupun baris. MongoDB hanya memiliki koleksi dan dokumen. Data yang disimpan dalam *database* MongoDB berupa file JSON yang disebut BSON (Binary JSON). MongoDB memiliki keunggulan dalam kinerja dan penggunaan sumber daya komputer yang lebih efisien. Dibandingkan dengan MySQL, MongoDB menggunakan lebih sedikit *processor* dan memori [10].

Pada MongoDB, terdapat fitur *Text Indexes* yang berfungsi untuk mendukung fungsi pencarian teks pada konten *string*. *Text Indexes* ini dapat mempercepat pencarian data dan dapat digunakan tanpa menggunakan *dependency* pihak ketiga [5]. Pada umumnya, *primary key* selalu dijadikan acuan untuk melakukan pencarian data yang cepat dan tepat. Berbeda dengan *Text Indexes* yang mengubah seluruh data yang bertipe teks / *string* kedalam bentuk indeks sehingga proses pencarian data yang banyak dapat berjalan secara cepat.

Bootstrap adalah suatu *framework* CSS yang mempermudah dan mempercepat pembuatan aplikasi web dan situs web yang responsif. Bootstrap juga menyediakan fitur untuk memudahkan pembuatan *layout* halaman web yang rapi dan mudah, serta memungkinkan modifikasi tampilan HTML dasar untuk menciptakan kesatuan desain pada seluruh halaman web yang dikembangkan dengan menggunakan komponen lain dari Bootstrap [11]. Bootstrap juga memiliki *plugin* jQuery yang dapat menghasilkan berbagai jenis komponen UI (*User Interface*) [12].

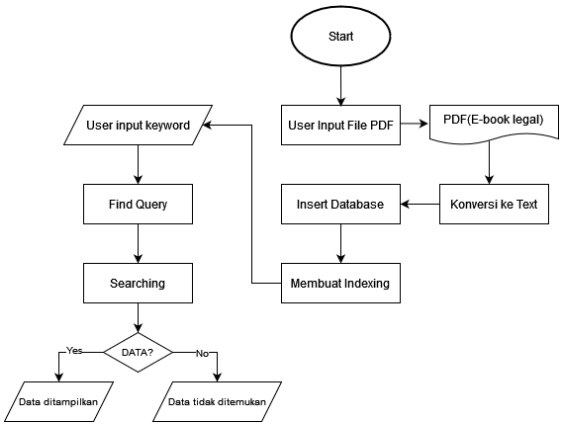
2. METODE PENELITIAN

Penelitian ini terbagi ke dalam beberapa tahapan, yaitu Identifikasi Masalah, Studi Pustaka, Analisis Metode, Perancangan sistem, Implementasi dan Analisis, dan Laporan Hasil Penelitian.



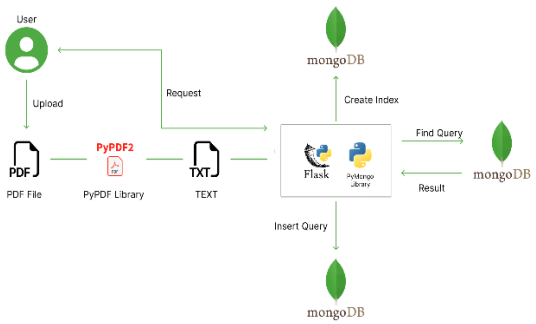
Gambar 1. Metode Penelitian

Pada Gambar 1 dijabarkan beberapa tahapan dalam melakukan penelitian ini. Tahap pertama yaitu identifikasi masalah. Identifikasi masalah dilakukan untuk menyelesaikan permasalahan yaitu jumlah data yang besar dan kecepatan pada fitur pencarian data. Setelah teridentifikasi, maka masuk ketahapan Studi Pustaka. Pada tahap ini Pada tahap ini dilakukan pengumpulan studi literatur berupa jurnal yang digunakan untuk membantu penelitian ini. Literatur diambil melalui berbagai jurnal nasional. Jurnal – jurnal yang dikumpulkan berkaitan dengan berbagai metode dan algoritma yang berguna untuk mengatasi masalah yang ada pada penelitian ini. Pada tahapan selanjutnya, dilakukan analisis metode dan teknologi yang akan digunakan dalam penelitian ini. Hasil dari tahap ini adalah dengan menggunakan *text indexing* pada data dokumen menggunakan MongoDB. Dalam penerapan *text indexing*, MongoDB digunakan sebagai *database* untuk menyimpan data dokumen dalam bentuk teks. Adapun Flask yang digunakan sebagai *framework* untuk mengimplementasikan *text indexing* ke dalam bentuk web. Berikut gambaran alur sistem dalam bentuk *flowchart* :



Gambar 2. *Flowchart*

Gambar 2 menjelaskan alur dari sistem yang dirancang dalam penelitian ini. Proses dimulai dengan *user* yang melakukan *upload file* dalam format dokumen pdf. Kemudian, *file* tersebut dikonversikan menjadi teks menggunakan *library* PyPDF2. Hasil teks dari proses konversi tersebut kemudian dimasukkan ke dalam *database* MongoDB. Setelah itu, dilakukan pembuatan indeks pada *field* tertentu yang akan dijadikan indeks untuk pencarian. Setelah pembuatan indeks selesai, *user* dapat melakukan pencarian kata kunci pada data yang ingin dicari. Kata kunci yang ditemukan akan diolah dalam *query* pencarian, sehingga data yang sesuai dengan kata kunci tersebut dapat ditemukan. Data yang ditemukan akan ditampilkan dalam bentuk web.



Gambar 3. Diagram Arsitektur

Gambar 3 merupakan gambaran keseluruhan sistem dalam bentuk diagram arsitektur. Selanjutnya tahap implementasi dan analisis. Pada tahap ini dilakukan implementasi sistem *text indexing* pada data dokumen berbasis web berdasarkan analisis dan rancangan sistem yang sudah dilakukan.

Kode Program 1. Kode Program Koneksi *Database*

Kode Program
import pymongo myclient=pymongo.MongoClient("mongodb://localhost:27017/") mydb=myclient["mongodbVSCodePlaygroundDB"] mycol = mydb["filedata"]

Kode Program 1 merupakan kode program untuk melakukan koneksi ke MongoDB dengan menggunakan library PyMongo. PyMongo memudahkan penggunaan MongoDB pada Python dengan menyediakan antarmuka yang mudah digunakan dan memungkinkan pengguna untuk melakukan operasi dasar seperti create, read, update, dan delete data pada MongoDB [13].

Kode Program 2. Kode program untuk *create index*

Kode Program
mydb.filedata.create_index([(“data”,"text")])

Kode Program 2 merupakan *query create index*. Dalam penelitian ini, dilakukan pembuatan indeks pada *field* "data" dalam *collection* "*filedata*”. Indeks ini dibuat dengan menggunakan metode "*create_index*" dan ditentukan sebagai indeks berjenis "*text*". Penggunaan indeks berjenis "*text*" memungkinkan untuk melakukan pencarian *keyword*

berjenis teks pada data yang tersimpan dalam *field* "data". Dengan demikian, operasi pencarian data akan menjadi lebih cepat dan efisien. Setelah tahap implementasi selesai, dilakukan analisis terhadap perbandingan kecepatan pencarian data dengan metode *text indexing* dengan metode pencarian data biasa. Pada tahapan terakhir, dilakukan penulisan laporan hasil penelitan dari tahap awal hingga tahap akhir penelitian.

3. HASIL DAN PEMBAHASAN

3.1. Penelitian Terdahulu

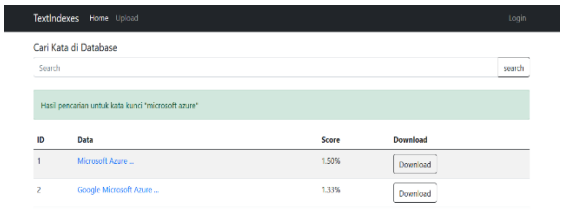
Pada penelitian yang berjudul "Implementasi *Full Text Search* Pada Sistem Informasi Perpustakaan Menggunakan Laravel" membahas tentang membangun aplikasi pencarian data buku di perpustakaan untuk mempercepat proses pencarian dengan menggunakan *text indexing*. *Text indexing* mengubah data *string* menjadi indeks sehingga bisa mempercepat proses pencarian data buku pada perpustakaan tersebut. Teknologi yang digunakan dalam penelitian ini adalah Elastic search, SQL, Nginx dan Laravel [14].

Pada penelitian yang berjudul "Penggunaan *Fulltext Indexing* Untuk Meningkatkan Efisiensi Pencarian Data Pada Basis Data MYSQL" membahas uji coba pencarian data pada tabel yang diberi indeks dan yang belum diberi indeks. Teknologi yang digunakan pada penelitian ini adalah MySQL *database*. Penelitian tersebut menerapkan fitur *text indexing* yang ada pada *database* MySQL. Pengujian dilakukan dengan perbandingan antara tabel yang diberi indeks dengan yang tidak diberi indeks. Pada tabel yang diberi indeks akan menggunakan fungsi MATCH AGAINTS pada MySQL dan tabel yang tidak diberi index akan menggunakan fungsi LIKE pada MySQL [15].

Pada penelitian yang berjudul "Pengembangan Sistem Pencarian Karya Akhir Berdasarkan Abstrak Menggunakan *Full-Text Searching* Di Sistem Informasi Perpustakaan Jurusan Teknik Elektro Universitas Negeri Jakarta" membahas pengembangan sistem pencarian yang dapat mencari dengan cepat tetapi tetap relevan untuk membantu mahasiswa dalam mencari tugas akhir berdasarkan abstrak. Teknologi yang digunakan pada penelitian tersebut adalah PHP dan MySQL. Penelitian tersebut menerapkan fitur *text indexing* pada MySQL ke dalam 229 abstrak tugas akhir yang kemudian dilakukan pencarian Tugas Akhir berdasarkan abstrak. Penerapan fitur *text indexing* memiliki tujuan untuk mempercepat proses pencarian tetapi tetap relevan. Pengujian dilakukan dengan menggunakan fungsi LIKE, WHERE, ORDER, MATCH dan AGAINTS pada MySQL. Hasil akhir dari penelitian tersebut berupa perbandingan relevansi antar fungsi[16].

3.2. Hasil dan Pembahasan

Dalam penelitian ini, menghasilkan rancangan sistem dan analisis perbandingan terhadap implementasi teknik *text indexing* dalam MongoDB. Rancangan sistem dalam bentuk web dibuat untuk membantu proses *insert data* dan pencarian data pada penelitian ini. Web dirancang dengan menggunakan *framework* Flask dengan Bootstrap sebagai UI(*User Interface)*. Sebanyak 5000 *record* data telah dimasukkan, dengan rata-rata setiap *record* terdiri dari teks yang memiliki jumlah kata antara 25.000 hingga 125.000 kata. Data tersebut berasal dari ekstraksi teks dokumen pdf *e-book* yang diambil dari sumber *website* legal nasional.



Gambar 4. Tampilan UI Pencarian Data

Gambar 4 merupakan tampilan UI untuk pencarian data berdasarkan kata kunci. Data yang ditemukan akan ditampilkan pada tampilan UI tersebut.

Kode Program 3. Kode Program Pencarian Data

Kode Program

word = request.form['search']
# Tanpa index 	
mycol.find({"data": {"$regex": word}})
.explain()
# Dengan index 	
mycol.find({"$text": {"$search": word}})
.explain()

Kode Program 3 merupakan kode program untuk pencarian data. Kode program tersebut terhubung dengan UI yang telah dibuat. Terdapat 2 perbedaan kode program pencarian yang digunakan. Pada pencarian tanpa indeks, menggunakan operator $regex yang berfungsi untuk pencarian data biasa dan *field* "data" merupakan *field* yang dipilih untuk dilakukan pencarian. Pada pencarian dengan Indeks, menggunakan operator $search untuk menentukan kata yang ingin dicari dan operator $*text* untuk menentukan bahwa kolom yang dicari haruslah kolom teks yang diindeks. Pada masing – masing kode program diakhiri dengan fungsi *explain*() yang berfungsi untuk menampilkan waktu eksekusi *query* yang dilakukan untuk menemukan data. Dengan fungsi tersebut, dihasilkan perbandingan kecepatan eksekusi *query* pada tabel berikut.

Tabel 1. Hasil Kecepatan Eksekusi *Non Index*

Non Index			
	1 Keyword	2 Keyword	3 Keyword
Jumlah Record	lama eksekusi (ms)	lama eksekusi (ms)	lama eksekusi (ms)
100	48	274	263
200	112	496	508
300	133	670	689
400	175	949	999
500	202	1191	1196
…	…	…	…
1100	449	2501	2492
1200	456	2695	2692
1300	494	2973	3071
1400	519	3091	3115
1500	584	3672	3617
...	…	…	…
3100	1210	6871	7034
3200	1198	7133	6884
3300	1209	7307	7365
3400	1243	7475	7300
3500	1239	7637	7767
...	…	…	…
4600	1897	10719	10398
4700	1869	10202	10346
4800	1988	10705	10437
4900	1894	10883	10850
5000	1990	10950	11080

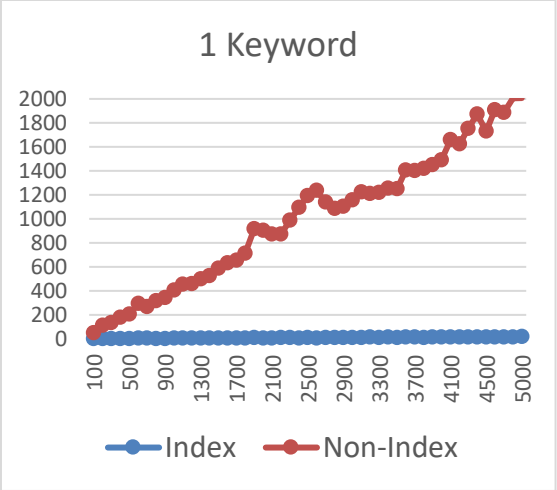
Tabel 1 merupakan hasil kecepatan eksekusi *query* pada pencarian kata kunci tanpa indeks. Lama eksekusi memiliki satuan milisekon (ms). Dapat dilihat bahwa lama eksekusi tanpa indeks cenderung naik ketika jumlah *record* bertambah banyak. Hasil kecepatan *query* pada pencarian kata kunci dengan indeks dapat dilihat pada Tabel 2 dibawah ini.

Tabel 2. Hasil Kecepatan Eksekusi *Index*

Index			
	1 Keyword	2 Keyword	3 Keyword
Jumlah Record	lama eksekusi (ms)	lama eksekusi (ms)	lama eksekusi (ms)
100	0.1	1	1
200	1	1	2
300	1	1	2
400	2	2	3
500	2	2	3
…	…	…	…
1100	4	4	6
1200	4	3	6
1300	4	5	6
1400	5	5	7
1500	5	4	7
...	…	…	…
3100	11	8	14
3200	12	8	17
3300	10	10	16
3400	12	11	17
3500	10	11	16
...	…	…	…
4600	12	14	21
4700	15	14	25
4800	16	15	23
4900	14	15	25
5000	17	18	25

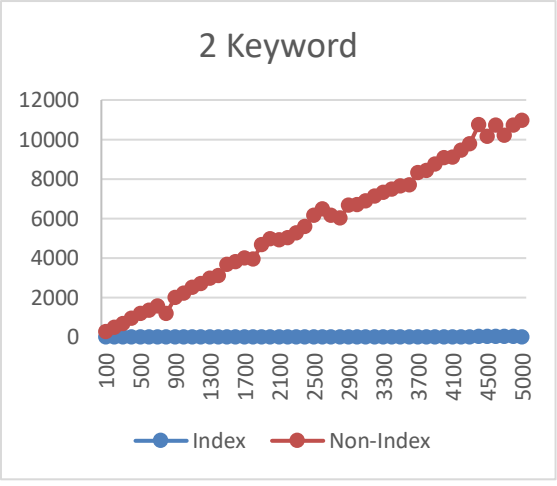
Pada Tabel 2, terlihat bahwa lama eksekusi tidak terlalu berpengaruh oleh pertambahan *record* data. Terlihat bahwa pada jumlah *record* 1100, 1200, dan 1300 dengan pencarian *query* 1 *keyword* memiliki lama eksekusi yang sama. Pada jumlah *record* 3100 sampai dengan 3500 memiliki lama eksekusi yang cenderung naik turun berbeda dengan Tabel 1 yang cenderung naik signifikan.

Hasil penelitian juga dibagi menjadi 3 percobaan perbandingan melalui pencarian kata 1 *keyword*, 2 *keyword*, dan 3 *keyword*. Pada pencarian 1 *keyword*, dilakukan percobaan pencarian hingga *record* data ke 5000. Pada record ke 5000 didapatkan perbandingan 1990ms dengan 17ms dimana lama eksekusi pada pencarian dengan indeks memiliki kecepatan hingga 11705,88% lebih cepat dibandingkan pencarian tanpa indeks. Grafik perbandingan dapat dilihat dibawah ini.



Gambar 5. Grafik Perbandingan 1 *Keyword*

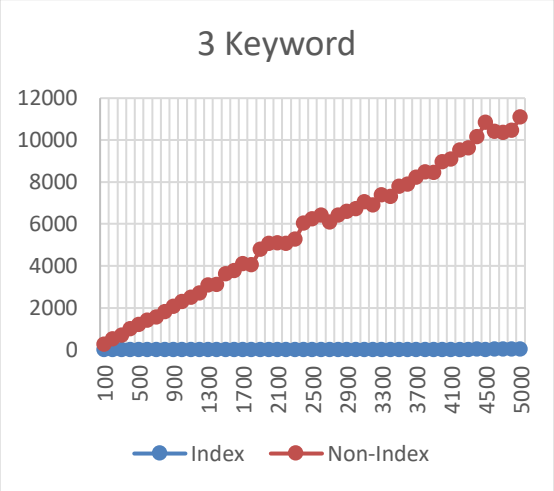
Gambar 5 merupakan grafik perbandingan pencarian 1 *keyword*. Terlihat bahwa lama pencarian data tanpa indeks meningkat seiring meningkatnya jumlah data *record*, sedangkan pencarian dengan indeks terlihat lebih stabil. Grafik perbandingan 2 *keyword* dapat dilihat dibawah ini.



Gambar 6. Grafik Perbandingan 2 *Keyword*

Gambar 6 merupakan grafik perbandingan pencarian 2 *keyword*. Terlihat bahwa perbandingan kecepatan sama seperti pada Gambar 5, dimana lama

pencarian tanpa indeks meningkat dibandingkan pencarian indeks. Pada record ke 5000 didapatkan perbandingan 10950ms dengan 18ms dimana lama eksekusi pada pencarian dengan indeks meningkat hingga 60833,33% lebih cepat dibandingkan pencarian tanpa indeks. Grafik perbandingan 3 *keyword* dapat dilihat dibawah ini.



Gambar 7. Grafik Perbandingan 3 *Keyword*

Gambar 7 merupakan grafik perbandingan pencarian 3 *keyword*. Terlihat sama seperti pada Gambar 5 dan Gambar 6 dimana lama pencarian tanpa indeks tetap meningkat seiring data *record* bertambah. Pada record ke 5000 didapatkan perbandingan 11080ms dengan 25ms dimana lama eksekusi pada pencarian dengan indeks meningkat hingga 44320% lebih cepat dibandingkan pencarian tanpa indeks. Dari Gambar 5 sampai 7 dapat disimpulkan bahwa pencarian indeks memiliki kecepatan pencarian yang jauh lebih cepat dibandingan pencarian tanpa indeks.

4. DISKUSI

Penelitian sebelumnya membahas tentang pengembangan sistem pencarian karya akhir berdasarkan abstrak menggunakan *full-text searching* di sistem informasi perpustakaan Jurusan Teknik Elektro Universitas Negeri Jakarta dengan teknologi PHP dan MySQL. Tujuan dari penelitian tersebut adalah untuk mempercepat proses pencarian tetapi tetap relevan. Hasil dari penelitian tersebut adalah perbandingan relevansi antar fungsi. Sementara itu, hasil penelitian ini membahas tentang pengembangan sistem pencarian dengan teknik *text indexing* pada MongoDB dengan perbandingan kecepatan indeks dan tanpa indeks. Penelitian ini menggunakan 5000 *record* data dengan rata-rata setiap *record* terdiri dari teks yang memiliki jumlah kata antara 25.000 hingga 125.000 kata. Data tersebut berasal dari ekstraksi dokumen pdf *e-book* yang diambil dari sumber website legal nasional. Hasil penelitian menunjukkan bahwa penggunaan indeks dapat meningkatkan kecepatan pencarian secara signifikan. Lama eksekusi pada pencarian dengan indeks memiliki kecepatan hingga 11705,88% lebih cepat dibandingkan pencarian tanpa indeks pada pencarian 1 keyword. Pada pencarian 2 keyword dan 3 keyword, lama eksekusi pada pencarian dengan indeks meningkat hingga 60833,33% dan 44320% lebih cepat dibandingkan pencarian tanpa indeks. Kedua penelitian ini memiliki tujuan yang sama yaitu untuk mempercepat proses pencarian tetapi tetap relevan. Namun, teknologi yang digunakan berbeda yaitu MySQL pada penelitian sebelumnya dan MongoDB pada penelitian ini. Hasil penelitian ini menunjukkan bahwa penggunaan indeks pada MongoDB dapat meningkatkan kecepatan pencarian secara signifikan. Secara keseluruhan, hasil penelitian ini dapat memberikan kontribusi pada pengembangan sistem pencarian dengan teknik text indexing pada MongoDB. Namun, untuk dapat membandingkan keunggulan antara teknologi yang digunakan, dibutuhkan penelitian lebih lanjut dengan menggunakan teknologi yang sama sehingga hasilnya dapat dibandingkan secara objektif.

5. KESIMPULAN

Berdasarkan hasil penelitian dan diskusi, dapat disimpulkan bahwa pemberian fungsi *text indexing* MongoDB pada proses pencarian kata dalam bentuk *string*/teks dapat mempercepat lama pencarian. Dengan jumlah 5000 *record* data, lama eksekusi *query* pencarian dipersingkat hingga 11705,88% pada pencarian 1 *keyword*, 60833,33% pada pencarian 2 *keyword*, dan 44320% pada pencarian 3 *keyword*. Namun, untuk dapat membandingkan keunggulan antara text indexing MongoDB dengan MySQL, dibutuhkan penelitian lebih lanjut mengingat setiap teknologi memiliki kelebihannya masing – masing dalam beberapa kasus sehingga hasilnya dapat dibandingkan secara objektif.

Sebagai saran untuk penelitian selanjutnya, dapat dilakukan pengujian terhadap lama waktu yang dibutuhkan dalam melakukan insert data ke dalam database MongoDB. Informasi yang diperoleh dari hasil penelitian selanjutnya dapat membantu pengembang dalam memilih metode dan teknologi yang tepat untuk menangani jumlah data yang besar dan mempercepat pencarian pada MongoDB.

DAFTAR PUSTAKA

[1] S. Akbar, F. Fadhila, V. A. Saputro, E. Utami & Khusnawi, “Perbandingan Performa SQL dan NoSQL Dengan PHP Pada 5 Juta Data,” Indonesian Journal on Computer and Information Technology, vol. 6, no.1, pp. 38-42, 2021.

[2] M. F. Abdi, A. Susanto & Kusnawi, “Perbandingan Kecepatan Pencarian Data Sql Dan NoSQL,” Jurnal Teknologi Informasi, vol.5, no.1, 2021.

[3] M. Silalahi & D. Wahyudi, “Perbandingan

Performansi Database Mongodb Dan Mysql Dalam Aplikasi File Multimedia Berbasis Web," Computer Based Information System Journal, vol. 6, no. 1, pp. 38-42, 2018.

[4] A. Sonita & M. Sari, "Implementasi Algoritma Sequential Searching Untuk Pencarian Nomor Surat Pada Sistem Arsip Elektronik," Jurnal Pseudocode, vol. 5, no. 1 , 2018.

[5] MongoDB, "Welcome to the MongoDB Documentation - Text Indexes," 2022, https://www.mongodb.com/docs/manual/core/index-text/#text-indexes (accessed Okt. 18, 2022).

[6] MongoDB, " Text Search Languages," 2023, https://www.mongodb.com/docs/manual/reference/text-search-languages/#std-label-text-search-languages (accessed 27/03/2023)

[7] T. M. Kadarina & M. H. I. Hajar, "Pengenalan Bahasa Pemrograman Python Menggunakan Aplikasi Games Untuk Siswa/I Di Wilayah Kembangan Utara," Jurnal Abdi Masyarakat,vol. 5, no. 1, 2019.

[8] Novindri, G. F., & Saian, P. O. N., "Implementasi Flask pada Sistem Penentuan Minimal Order untuk Tiap Item Barang di Distribution Center pada PT XYZ Berbasis Website," Jurnal Mnemonic, vol. 5, no. 2, pp. 80-85, 2022.

[9] R. Irsyad, "Penggunaan Python Web Framework Flask Untuk Pemula," 2018, doi: 10.31219/osf.io/t7u5r.

[10] Renaldi, B. C. Santoso, Y. Natasya, S. Willian, & F. Alfando, "Tinjauan Pustaka Sistematis terhadap Basis Data MongoDB," JII: Jurnal Inovasi Informatika Universitas Pradita, vol 5, no. 2, pp. 132-142, 2020.

[11] Putra, M. Y., Safitri, N., Fauziah, N. F., Safei, A., Rayhan, & Lolly, W. R., "Desain Web Bagi Pemula Menggunakan Framework Bootstrap pada SMK Taruna Bangsa Bekasi," Jurnal Buana Pengabdian, vol 3, no. 1,pp. 134-148, 2021.

[12] Putra, M. Y., "Responsive Web Design Menggunakan Bootstrap Dalam Merancang Layout Website," Information System For Educators And Professionals, vol. 5, no. 1, pp. 61-70, 2020.

[13] PyMongo, "PyMongo 4.3.3 Documentation," 2023,https://pymongo.readthedocs.io/en/stable/ (accessed Feb. 02, 2023).

[14] I. Asyahri & M. S. Mauludin, "Implementasi Full Text Search Pada Sistem Informasi Perpustakaan Menggunakan Laravel," Informatika dan RPL, vol. 1, no. 1, pp. 1-9, 2019

[15] A. Hajar, E. Utami, & H. A. Fatta, "Penggunaan Fulltext Indexing Untuk Meningkatkan Efisiensi Pencarian Data Pada Basis Data MYSQL," Jurnal SISFOTENIKA, vol. 12, no. 2, pp. 213-222, 2022

[16] E. Selviyanti, H. Ajie & Widodo, "Pengembangan Sistem Pencarian Karya Akhir Berdasarkan Abstrak Menggunakan Full-Text Searching Di Sistem Informasi Perpustakaan Jurusan Teknik Elektro Universitas Negeri Jakarta," JTIM : Jurnal Teknologi Informasi dan Multimedia, vol. 1, no. 2, pp 85- 95, 2019.