JURNAL SISFOTENIKA
Vol. 12 No. 2, Juli 2022
p-ISSN: 2087-7897 ; e-ISSN : 2460-5344
DOI : 10.30700/jst.v12i2.1264

Penggunaan Fulltext Indexing Untuk Meningkatkan Efisiensi Pencarian Data Pada Basis Data MYSQL

Use of Fulltext Indexing to Improve Data Search Efficiency in MYSQL Databases

Ahmad Hajar*[1], Ema Utami[2], Hanif Al Fatta[3]
[1]Universitas Amikom Yogyakarta; Jl. Ring Road Utara, Ngringin, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281, (0274) 884201
[2,3] Dosen pembimbing, Universitas Amikom Yogyakarta
e-mail: ***[1]ahmad.1294@students,amikom.ac.id**, [2]ema.u@amikom.ac.id,
[3]hanif.a@amikom.ac.id

Abstrak

Proses pencarian data adalah salah satu kebutuhan dari sistem yang sangat dibutuhkan. Dengan semakin banyaknya data, maka proses pencarian akan semakin lama. Ada banyak cara untuk mempercepat proses pencarian data, salah satunya dengan memanfaatkan fitur Fulltext index yang ada di basis data MySQL. Pada penelitian ini akan dilakukan uji coba query pencarian pada tabel yang sudah diberi index dan pada tabel yang belum diberi index. Peneliti menggunakan operator like untuk melakukan pencarian data pada tabel yang tidak diberi index. Sedangkan pada tabel yang diberi index, peneliti menggunakan pencarian fulltext dengan sintak match against. Hasil penelitian ini menunjukkan bahwa pencarian fulltext pada tabel yang sudah diberi index lebih cepat dibandingkan dengan tabel yang tidak diberi index. Pada saat jumlah data ada 466.000, pencarian di tabel non index membutuhkan waktu 2.638,97 milisekon. Sedangkan pada tabel yang diberi index membutuhkan waktu 89,14 milisekon. Kemudian dampak dari pemberian index pada tabel adalah proses insert menjadi lebih lama dibandingkan pada tabel yang tidak diberi index. Pada tabel product_index ketika datanya ada 466.000, lama insertnya adalah 6.560,69 milisekon. Sedangkan pada tabel product_non_index adalah 420,96 milisekon. Pada penelitian selanjutnya diharapkan mencari metode pengetesan query yang lain serta mencari tahu akurasi antara operator like dan match against.

Kata kunci*— Query Testing, Query Speed Test, Fulltext Index, Mysql, Laravel*

Abstract

The process of searching for data is one of the needs of the system that is needed. With the more data, the search process will take longer. There are many ways to speed up the data search process, one of which is by utilizing the Full text index feature in the MySQL database. In this study, a search query will be tested on tables that have been indexed and on tables that have not been indexed. Researchers use the like operator to search for data on tables that are not indexed. Meanwhile, in the table given the index, the researcher uses a full text search with the match against syntax. The results of this study indicate that the full text search on a table that has been given an index is faster than a table that is not indexed. When the number of data is 466,000, searching in a non-index table takes 2,638.97 milliseconds. Meanwhile, the indexed table takes 89.14 milliseconds. Then the impact of giving an index to the table is that the insert process takes longer than the table that is not indexed. In the product index table when the data is 466,000, the insert time is 6,560.69 milliseconds. While the product non index table is 420.96

milliseconds. In future research, it is expected to look for other query testing methods and find out the accuracy between like operator and match against.

Keywords— *Query Testing, Query Speed Test, Fulltext Index, Mysql, Laravel*

1. PENDAHULUAN

Pencarian data adalah sesuatu proses yang sangat penting dalam sebuah sistem. Ketika sistem sudah mempunyai banyak data, terkadang proses pencarian data menjadi lebih lambat. Ada beberapa cara untuk mempercepat pencarian data, tergantung basis data yang digunakan. Chaitanya et al., (2019) mengimplementasikan B+ tree dengan inverted index untuk mempercepat pencarian data pada basis data PostgreSQL [1].

Pada basis data MongoDB, Kelec et al., (2019) melakukan text indexing tanpa dependency tambahan walaupun hasilnya kurang begitu bagus jika dibandingkan dengan menggunakan sistem pihak ketiga untuk pencarian fulltext [2]. Dan salah satu masalah yang didapatkan ketika melakukan text indexing pada basis data MongoDB adalah duplikasi data.

Pada basis data MySQL sendiri terdapat fitur indexing yang bisa digunakan untuk mempercepat pencarian data. Selviyanti et al., (2019) mengimplementasikan fulltext index pada basis data MySQL di kolom abstract untuk mempercepat pencarian karya akhir berdasarkan abstract [3]. Kemudian Wirawan et al., (2019) mengimplementasikan fulltext index dengan Artificial intelligence markup language untuk membuat chatbot di aplikasi Line Messeger yang bisa memberi jawaban yang cepat, relevan dan berasa seperti berkomunikasi dengan sejarawan [4].

Candra et al., (2020) juga mengimplementasikan fulltext index pada basis data MySQL dengan mode boolean untuk membuat chatbot di aplikasi Telegram [5]. Chatbot yang dibuat dipergunakan untuk membalas pesan secara otomatis bagi pengguna yang ingin mencari tahu informasi tentang STMIK PGRI Tangerang dengan hasil yang cepat dan relevan.

Hryhorova et al., (2018) mengimplementasikan fulltext di sistem e-learning yang sudah ada untuk mempercepat pencarian data pada semua topik pembelajaran [6]. Walaupun dibagian akhir mereka menyarankan untuk implementasi fulltext search dilakukan memakai sistem pihak ketika seperti Apache Solr atau Elasticsearch.

Pada basis data MySQL ada beberapa macam index seperti index, unique, primary dan fulltext [7]. Masing-masing index mempunyai kegunaan yang hampir sama tetapi memiliki perbedaan. Misal pada index primary dan unique tidak memperbolehkan ada data yang sama ditiap barisnya, perbedannya pada index unique bisa dipakai dibeberapa kolom sekaligus, sedangkan index primary hanya bisa digunakan disatu kolom.

Index index bisa digunakan pada beberapa kolom, dan tidak peduli apakah datanya kosong atau tidak. Untuk yang index fulltext digunakan untuk pencarian fulltext dengan menggunakan fungsi match dan against. Dan di dalam index fulltext terdapat 3 mode yaitu natural language, boolean, dan query expansion.

Pamungkas (2018) melakukan optimasi basis data MySQL dengan mengimplementasikan beberapa index seperti primary key, foreign key, dan index [8]. Rinartha et al., (2018) melakukan komparasi kecepatan antara MySQL pattern matching, Levenshtein distance, Jaccard Similarity dan MySQL fulltext Index [9], hasilnya yang tercepat adalah MySQL pattern matching tetapi tidak akurat, dan yang paling akurat dan dengan waktu proses yang cepat dibanding Levenshtein distance dan Jaccard Similarity adalah MySQL fulltext Index.

Pratama et al., (2018) melakukan analisa perbandingan query pencarian menggunakan fungsi match against pada mysql dengan tabel kamus [10], hasilnya pencarian dengan tabel kamus memberi hasil pencarian yang lebih baik namun menambah waktu proses pencarian.

Menurut Koszalka et al., (2005) efisiensi adalah kecepatan pemrosesan query yang dinyatakan dalam transaksi per detik [11]. Semakin cepat proses query maka semakin efisien query tersebut.

Dari uraian diatas, dengan pertimbangan index fulltext adalah salah fitur yang tersedia dibasis data MySQL, yang bisa kita gunakan tanpa perlu install dependency pihak ketiga, dan dari beberapa penelitian menunjukan bahwa index fulltext dapat mempercepat pencarian data, maka peneliti akan meneliti tentang index fulltext dan melihat seberapa peningkatan kecepatan query antara yang sudah terindex dan yang belum. Dan melihat adakah dampak negatif dari pemberian index pada tabel. Penelitian akan dilakukan pada basis data MySQL versi 8. Dan kecepatan query ditetapkan dalam millisecond. Semakin kecil nilainya berarti semakin cepat atau efisien.

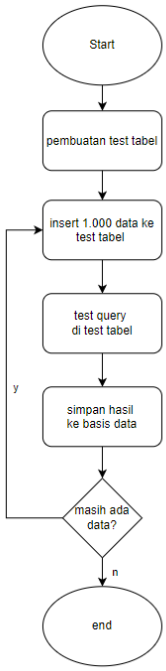
2. METODE PENELITIAN

2.1 Gambaran umum obyek penelitian dan persiapan data

Basis data MySQL mempunyai fitur Fulltext search yang berguna untuk mempercepat proses pencarian data. Pada penelitian kali ini, penulis mencoba untuk membandingkan lama proses pencarian data pada tabel yang sudah diberi index dan yang belum. Data yang digunakan adalah data product sederhana yang berisi: itemid, nama produk, deskripsi produk dan harga produk yang berjumlah 466.000 data.

2.2 Pengujian query

Pada tahap pengujian query ini, terdapat beberapa proses yang dilalui seperti pembuatan test tabel. Kemudian memasukkan data ke test tabel, lalu melakukan test query di test tabel dan menyiman hasilnya di basis data. Proses lengkapnya bisa dilihat pada gambar 1 dibawah ini



Gambar 1 Proses Pengujian Query

2.2.1 Pembuatan test tabel

Test tabel yang digunakan ada 2, yang pertama tabel product_non_index, yang akan digunakan untuk pengetesan pencarian menggunakan operator like. Yang kedua tabel product_index yang akan digunakan untuk pengetesan pencarian menggunakan metode match against. Struktur dari test tabel product_non_index bisa dilihat pada gambar 2 dibawah ini

#	Name	Type
1	itemid	varchar(50)
2	name	varchar(255)
3	description	longtext
4	price	varchar(255)

Gambar 2 struktur test tabel product_non_index

Untuk struktur tabel product_index bisa dilihat pada gambar 3 dibawah ini.

#	Name	Type
1	itemid	varchar(50)
2	name	varchar(255)
3	description	longtext
4	price	varchar(255)

Gambar 3 struktur tabel product_index

Pada tabel product_index, kolom yang diberi index adalah name dan description dengan menggunakan command: ALTER TABLE `product_index` ADD INDEX `product_index_id` (`name`, `description`);

2.2.2 Insert 1000 data ke test tabel

Setelah test tabel dibuat, penulis memulai pengetesan dengan cara mengambil 1000 data pada tabel products kemudian dimasukkan kedalam test tabel product_non_index dan tabel product_index.

2.2.3 Test query

Setelah test tabel terisi data, penulis menjalankan test query pencarian seperti pada gambar 4 dibawah.

```
switch($jumlah_keyword){
    case 1:
        $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where `name` like '%samsung%' or `description` like '%samsung%'";
        break;
    case 2:
        $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where `name` like '%samsung note%' or `description` like '%samsung note%'";
        break;
    case 3:
        $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where `name` like '%samsung note 9%' or `description` like '%samsung note 9%'";
        break;
    case 4:
        $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where `name` like '%samsung note 9 512gb%' or `description` like '%samsung note 9 512gb%'";
        break;
}
```

Gambar 4 test query pada tabel product_non_index

Seperti terlihat pada gambar diatas, terdapat 4 kali pengujian yaitu pengujian 1 keyword (“samsung”), 2 keyword (“samsung note”), 3 keyword (“samsung note 9”), dan 4 keyword

("samsung note 9 512gb"). Untuk query pencarian pada tabel product_index bisa dilihat pada gambar 5 dibawah.

```
if($tipe == "natural language") {
    switch($jumlah_keyword){
        case 1:
            $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where match(name, description) against ('samsung' in natural language mode)";
            break;
        case 2:
            $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where match(name, description) against ('samsung note' in natural language mode)";
            break;
        case 3:
            $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where match(name, description) against ('samsung note 9' in natural language mode)";
            break;
        case 4:
            $query = "SELECT `itemid`, `name`, `description`, `price` FROM `{$table}` where match(name, description) against ('samsung note 9 512gb' in natural language mode)";
            break;
    }
}
```

Gambar 5 test query pada tabel product_index

Karena peneliti menggunakan script untuk membantu proses pengetesan query, semua proses pengetesan query terjadi di background, dan datanya langsung disimpan di basis data.

2.2.4 Simpan hasil test ke dalam basis data

Dari hasil menjalankan test query, data yang didapat dimasukkan kedalam basis data. Proses akan diulangi lagi dari insert 1000 data sampai seluruh data yang ada di tabel products di copy ke test tabel. Contoh hasil dari pengujian pada tabel product_non index bisa dilihat pada tabel 1 dibawah.

Tabel 1 hasil test query pada tabel product_non_index

	1 keyword	2 keyword	3 keyword	4 keyword
Jumlah data	**Lama eksekusi (milisekon)**	**Lama eksekusi (milisekon)**	**Lama eksekusi (milisekon)**	**Lama eksekusi (milisekon)**
1000	6,67	6,63	6,64	5,83
2000	10,11	10,29	10	10,1
3000	15,79	16,61	16,53	16,69
4000	34,97	23,43	21,47	21,87
5000	28,13	36,39	28,12	27,2
…	…	…	…	…
10000	56,76	60,58	57,7	58,29
11000	69,08	61,09	59,69	68,38
12000	69,91	61,3	68,75	73,79
13000	74,58	74,07	75,04	76,95
14000	77,16	78,75	85,71	84,47
15000	78,73	86,26	85,13	82,13
…		…	…	…
460000	2469,7	2609,87	2629,7	2454,47
461000	2597,94	2470,02	2622,15	2589,29
462000	2572,24	2509,47	2510,83	2574,94
463000	2614,35	2550,28	2589,65	2630,76
464000	2533,98	2510,14	2513,25	2590,17
465000	2671,21	2629,02	2594,44	2556,56
466000	2487,03	2638,97	2655,21	2614,9

Lama eksekusi pada tabel 1 memiliki satuan milisekon. Bisa kita lihat bahwa pada tabel product_non_index, semakin banyak data semakin lama eksekusi query pencariannya. Hasil pengujian pada tabel product_index bisa dilihat pada tabel 2 dibawah.

Tabel 2 hasil test query pada tabel product_index

	1 keyword	2 keyword	3 keyword	4 keyword
Jumlah data	**Lama eksekusi (milisekon)**	**Lama eksekusi (milisekon)**	**Lama eksekusi (milisekon)**	**Lama eksekusi (milisekon)**
1000	5,38	4,86	4,41	4,91
2000	0,88	2,2	2,09	2,17
3000	2,96	6,49	2,39	2,36
4000	1,02	4,01	2,74	3,09

5000	1,03	3,51	3,38	3,65
…	…	…	…	…
10000	1,52	6,28	6,12	6,09
11000	1,42	7,13	6,67	7,22
12000	1,47	7,96	7,36	7,38
13000	2,04	8,56	8,18	8,13
14000	5,91	13,13	8,27	8,28
15000	1,86	9,83	8,82	8,87
…	…	…	…	…
460000	37,93	317,64	297,55	303,89
461000	39,57	313,23	313,27	310,53
462000	38,37	326,41	302,64	315,95
463000	41,04	319,2	296,25	310,1
464000	42,49	314,23	313,62	345,58
465000	45,42	307,33	314,51	303,43
466000	89,14	379,83	386,26	418,29

Bisa kita lihat bahwa pada tabel product_index, jumlah data tidak terlalu berpengaruh pada lama eksekusi query karena lama prosesnya naik turun seperti pada jumlah data 13000, 14000, dan 15000 dengan 1, 2 dan 3 keyword. Bisa kita lihat bahwa lama eksekusinya naik turun tetapi tidak terlalu signifikan berbeda dengan pengujian yang dilakukan pada tabel product_non_index yang cenderung naik terus.

3. HASIL DAN PEMBAHASAN

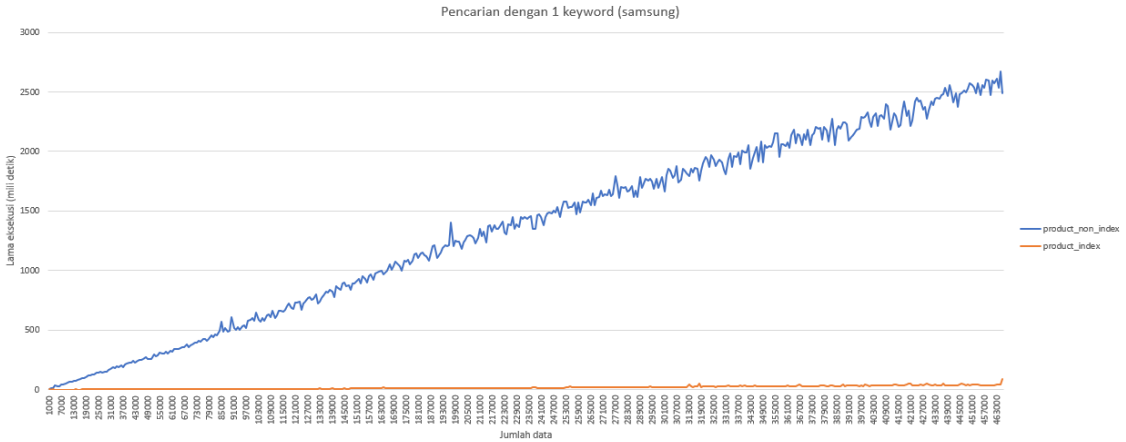
Dari tabel 1 dan 2 bisa kita lihat dengan adanya penambahan index, terjadi peningkatan proses pencarian ketika jumlah datanya 466.000 seperti terlihat pada tabel 3 dibawah.

Tabel 3 lama proses query ketika jumlah datanya 466.000

Tabel	1 keyword (milisekon)	2 keyword (milisekon)	3 keyword (milisekon)	4 keyword (milisekon)
Product_non_index	2.487,03	2.638,97	2.655,21	2.614,9
Product_index	89,14	379,83	386,26	418,29

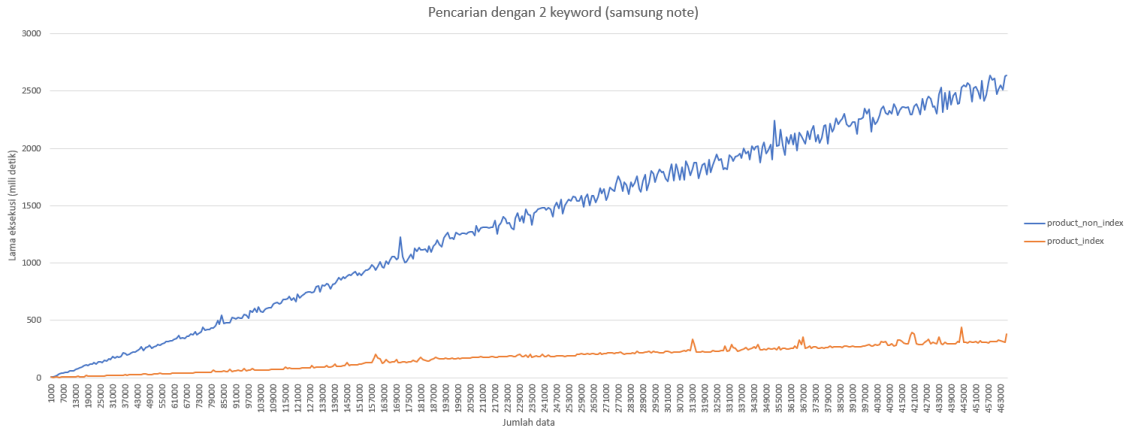
Pada saat pencarian 1 keyword, lama query di tabel product_non_index adalah 2.487,03 sedangkan pada tabel product_index adalah 89,14. Terjadi peningkatan sebesar 2.790,03% lebih cepat dibanding pencarian pada tabel product_non_index. Sebagai perbandingan, bisa kita lihat pada gambar 6 dibawah

.



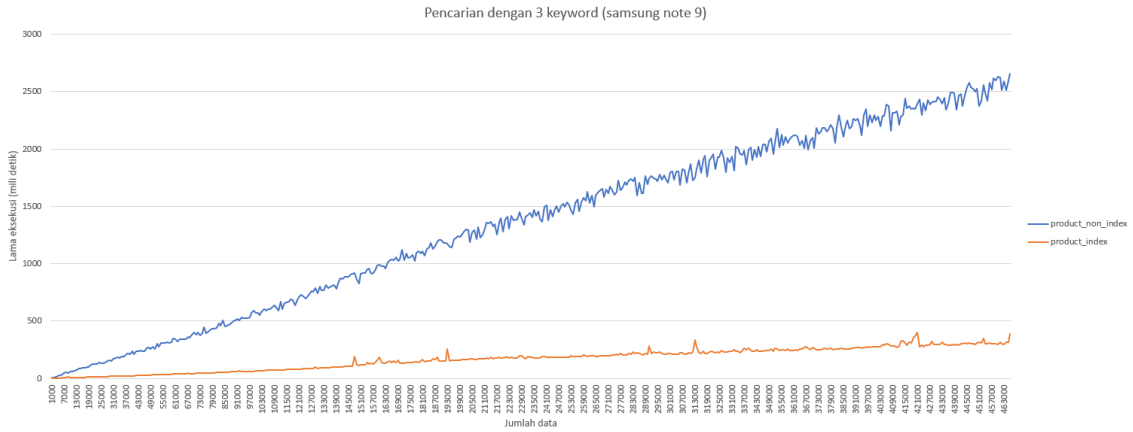
Gambar 6 lama eksekusi query 1 keyword

Bisa dilihat bahwa pencarian pada tabel product_non_index semakin menanjak ketika jumlah data semakin banyak, berbeda dengan pencarian pada tabel product_index yang jauh lebih landai. Pada pencarian 2 keyword bisa dilihat pada gambar 7 dibawah.



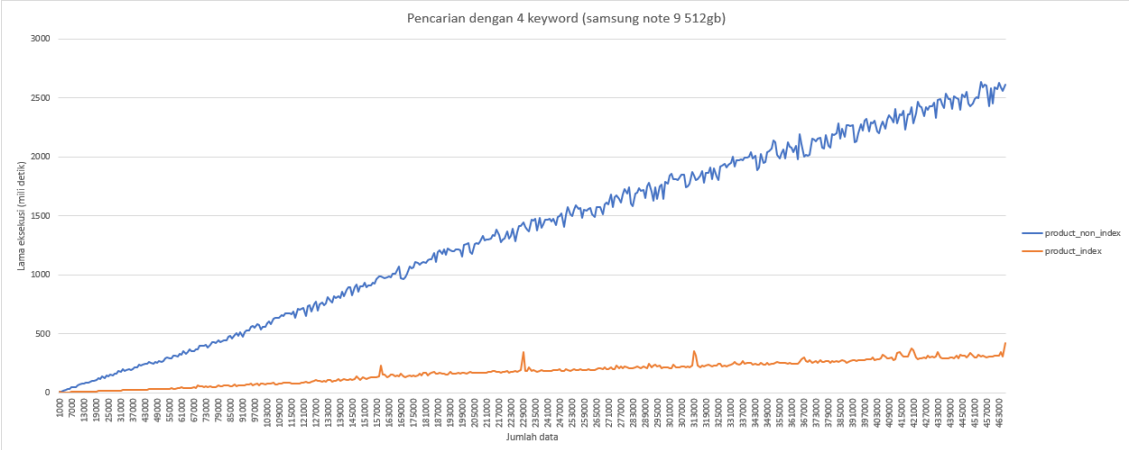
Gambar 7 lama eksekusi query 2 keyword

Seperti pada pencarian 1 keyword, lama eksekusi query pada tabel product_non_index semakin banyak datanya maka semakin lama proses eksekusi querynya. Pada pencarian 3 keyword bisa dilihat pada gambar 8 dibawah.



Gambar 8 lama eksekusi query 3 keyword

Seperti pada pencarian 1 dan 2 keyword, lama eksekusi query pada tabel product_non_index semakin banyak datanya maka semakin lama proses eksekusi querynya. Pada pencarian 4 keyword bisa dilihat pada gambar 9 dibawah.



Gambar 9 lama eksekusi query 4 keyword

Seperti pada pencarian 1, 2, dan 3 keyword. Lama eksekusi query pada tabel product_non_index semakin banyak datanya maka semakin lama proses eksekusi querynya. Dari gambar 6 sampai 9 bisa kita lihat, bahwa pencarian pada tabel product_index jauh lebih cepat dibanding pada tabel product_non_index.

Saat melakukan pengetesan query, peneliti juga mendokumentasikan lama proses insert data kedalam tabel product_non_index dan tabel product_index. Hasilnya bisa dilihat pada tabel 4 dibawah.

Tabel 4 lama insert data

	product_non_index	**product_index**
Jumlah data	**Lama insert (milisekon)**	**Lama insert (milisekon)**
1000	113,73	247,18
2000	26,61	144,78
3000	25,62	148,43
4000	41,54	157,9
5000	30,55	157,57
…	…	…
11000	31,38	137,68
12000	51,11	144,85
13000	34,28	151,63
14000	42,89	146,88
15000	40,31	150,78
…	…	…
460000	415,97	5259,86
461000	408,06	7507,04
462000	393,6	7313
463000	430,06	7218,29
464000	444,96	6705,67
465000	430,98	5787,11
466000	420,96	6560,69

Seperti terlihat pada tabel 4 bahwa lama insert pada tabel product_index jauh lebih lama dibandingkan pada tabel product_non_index.

4. KESIMPULAN

Kesimpulan yang dapat diambil dari penelitian ini adalah pemberian index pada tabel akan meningkatkan proses eksekusi query. Pada saat jumlah data 466.000, lama eksekusi query

pencarian 1 keyword pada tabel product_non_index adalah 2.487,03 milisekon, sedangkan pada tabel product_index adalah 89,14 milisekon. Terjadi peningkatan sebesar 2.790,03%, yang artinya pencarian pada tabel product_index jauh lebih efisien dibandingkan pencarian pada tabel product_non_index

Kemudian dampak dari pemberian index pada tabel adalah proses insert menjadi lebih lama dibandingkan pada tabel yang tidak diberi index. Pada tabel product_index ketika datanya ada 466.000, lama insertnya adalah 6.560,69 milisekon sedangkan pada tabel Product_non_index adalah 420,96 milisekon.

5. SARAN

Pada penelitian selanjutnya diharapkan mencari metode pengetesan query yang lain serta mencari tahu akurasi antara operator like dan match against.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

[1] B. Sri Sai Krishna Chaitanya, D. Ajay Kumar Reddy, B. Pavan Sai Eshwar Chandra, A. Bala Krishna, Remya R. K. Menon, 2019, *Full-text Search Using Database Index*: https://ieeexplore.ieee.org/abstract/document/9128683

[2] Aleksandar Kelec, Igor Dujlovic, Nikola Obradovic, 2019, *One approach for full-text search of files in MongoDB based systems*: https://ieeexplore.ieee.org/abstract/document/8717777

[3] Erna Selviyanti, Hamidillah Ajie, Widodo, 2019, *Pengembangan Sistem Pencarian Karya Akhir Berdasarkan Abstrak Menggunakan Full-Text Searching Di Sistem Informasi Perpustakaan Jurusan Teknik Elektro Universitas Negeri Jakarta*: http://journal.sekawan-org.id/index.php/jtim/article/view/8

[4] Kadek Teguh Wirawan, I Made Sukarsa, I Putu Agung Bayupati, 2019, *Balinese Historian Chatbot using Full-Text Search and Artificial Intelligence Markup Language Method*: http://www.mecs-press.net/ijisa/ijisa-v11-n8/IJISA-V11-N8-3.pdf

[5] Hokianto Candra, Rino, Riki, 2020, *Designing a Chatbot Application for Student Information Centers on Telegram Messenger Using Fulltext Search Boolean*: http://jurnal.kdi.or.id/index.php/bt/article/view/106

[6] T. A. Hryhorova, O. S. Moskalenko, 2018, *Enhanced access to training information in E-learning systems*: https://ieeexplore.ieee.org/abstract/document/8400124

[7] Mysql Team, *MySQL 8.0 Reference Manual - Including MySQL NDB Cluster 8.0*, https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/mysql-8.0-en.pdf diakses 21 Juni 2022

[8] Ridho Pamungkas, 2018, *Optimalisasi Query Dalam Basis Data My Sql Menggunakan Index*: https://www.researchgate.net/publication/325530763_Optimalisasi_Query_Dalam_Basis_Data_My_Sql_Menggunakan_Index

[9] Komang Rinartha, Wayan Suryasa, Luh Gede Surya Kartika. 2018, *Comparative Analysis of String Similarity on Dynamic Query Suggestions:* https://ieeexplore.ieee.org/abstract/document/8692996

[10] Zudha Pratama, Yans Safarid Hudha, M Lukman Prayoghi, 2018, *Analisa Perbandingan Query Pencarian Menggunakan Fungsi Match-Against Pada MySQL Dengan Tabel Kamus:* *https://media.neliti.com/media/publications/296405-analisa-perbandingan-query-pencarian-men-abff9275.pdf*

[11] I. Pozniak-Koszalka and M. Helwich. 2005, *Experimentation System for Evaluating MySQL Database Management System Efficiency*: https://www.actapress.com/Abstract.aspx?paperId=19052