Journal Sensi:Strategic Of Education in Information System
Accreditation rank 5
SK Dirjen DIKTI, RisTek No. 10/C/C3/DT.05.00/2025; E-ISSN: 2655-5298 P-ISSN: 2461-1409

Research Article **Open Access (CC–BY-SA)**

Optimalisasi Kinerja Indexing Dalam Pencarian Data Di Database Mongodb

Karunia Suci Lestari [1], Fadli Asyp Riansyah [2], Ahmad Herkal Taqyudin [3], Janu Ilham Saputro [4]
[1, 2, 3] Prodi Teknik Informatika Fakultas Sains dan Teknologi Universitas Raharja
[4] Prodi Sistem Informasi Fakultas Sains dan Teknologi Universitas Raharja
Email: [1] suci@raharja.info, [2] fadli.asyp@raharja.info, [3]herkal@raharja.info, [4] janu@raharja.info

Abstrak

Pengindeksan dalam MongoDB merupakan mekanisme krusial untuk mengoptimalkan kinerja query pada basis data non-relasional. Tanpa indeks yang tepat, sistem dipaksa melakukan collection scan yang membebani sumber daya. Penelitian ini mengulas konsep dasar pengindeksan, strategi implementasi yang efektif, serta praktik terbaik manajemen indeks. Menggunakan metodologi eksperimental, penelitian ini membandingkan performa query sebelum dan sesudah penerapan indeks. Hasil penelitian menunjukkan bahwa penggunaan indeks secara signifikan meningkatkan efisiensi pencarian, di mana waktu eksekusi query berhasil direduksi dari 580 ms menjadi 3 ms, atau mengalami peningkatan performa sebesar 99,4%. Temuan ini menegaskan bahwa pemahaman mendalam mengenai pola query dan struktur data sangat penting bagi pengembang untuk menjamin skalabilitas dan responsivitas aplikasi berbasis MongoDB.
Kata Kunci: *MongoDB, Pengindeksan, Database NoSQL, Kinerja Query, Skalabilitas.*

Abstract

Indexing in MongoDB is a crucial mechanism for optimizing query performance in non-relational databases. Without proper indexing, the system is forced to perform a collection scan, which consumes significant resources. This study examines the fundamental concepts of indexing, effective implementation strategies, and best practices for index management. Using an experimental methodology, this research compares query performance before and after index application. The results demonstrate that indexing significantly enhances search efficiency, with query execution time reduced from 580 ms to 3 ms, representing a 99.4% performance improvement. These findings emphasize that a deep understanding of query patterns and data structures is essential for developers to ensure the scalability and responsiveness of MongoDB-based applications.
Keywords: *MongoDB, Indexing, NoSQL Database, Query Performance, Scalability.*

1. PENDAHULUAN

Setiap hari, miliaran data diciptakan oleh kemajuan teknologi informasi. Oleh karena itu, jumlah data yang masuk ke *database* meningkat secara signifikan seiring berjalannya waktu. Ada kemungkinan bahwa jumlah data yang signifikan ini akan mempengaruhi seberapa cepat orang dapat mengakses *Database* [1]. Selama beberapa dekade, *database* relasional *(SQL)* telah

menjadi standar pemrosesan data. Di sisi lain, pengelolaan database seperti *NoSQL* (bukan *SQL*) telah berkembang dengan menggunakan metode baru seperti *key- value, Document-oriented,* dan *graph*.

Dalam ekosistem basis data modern, pengindeksan merupakan strategi krusial untuk mengatasi latensi kueri pada volume data yang masif. Optimasi indeks pada database berorientasi dokumen seperti MongoDB sangat bergantung pada pemilihan key yang tepat untuk meminimalkan *resource* contention [2]. Secara teknis, *MongoDB* menggunakan struktur *WiredTiger storage engine* yang mengelola indeks melalui arsitektur *B-tree* untuk mempercepat pencarian tanpa harus memindai seluruh koleksi (*collection scan*). Penelitian terbaru oleh Kumar dan Sharma (2023) menegaskan bahwa implementasi indeks yang tidak terukur dapat menyebabkan degradasi performa pada operasi penulisan (*write-heavy workloads*) karena adanya beban tambahan pada proses *update* indeks. Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi efektivitas berbagai tipe indeks dalam meningkatkan skalabilitas aplikasi, sekaligus memberikan panduan praktis manajemen indeks yang relevan dengan perkembangan fitur terbaru *MongoDB [3]*.

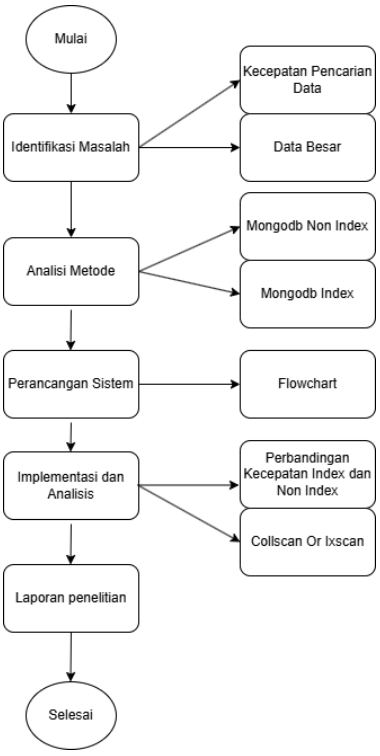
MongoDB adalah salah satu *database NoSQL* paling populer saat ini. *MongoDB* menyimpan data dalam format *BSON*, yang merupakan duplikat *JSON*, berbeda dengan *database SQL* yang menggunakan relasi tabel. *MongoDB* memiliki fitur indeks teks yang dapat digunakan dalam pencarian data yang terdiri dari banyak string atau teks untuk mendukung fungsi pencarian teks pada konten *string*. Peneliti akan menggunakan fitur *indexing* teks *MongoDB* untuk mengindeks data dokumen *PDF* yang telah diubah menjadi teks. Setelah itu, teks dimasukkan ke dalam database *MongoDB* sebelum diindeks. Selanjutnya, data yang sudah terindeks dan data yang belum terindeks dievaluasi berdasarkan kinerja *query* pencarian. *Database MongoDB* digunakan untuk membandingkan kecepatan [4]. Dengan menggunakan fitur *indexing* teks *MongoDB*, peneliti akan menerapkan *indexing* teks ke dalam data dokumen *PDF* yang telah diubah menjadi teks. Teks tersebut kemudian dimasukkan ke dalam *database MongoDB* sebelum diindeks. Setelah itu, performa *query* pencarian dibandingkan dengan data yang sudah terindeks dan data yang belum terindeks. Kecepatan perbandingan dilakukan dengan database *MongoDB*. Untuk mendukung fungsi pencarian teks pada konten string, ada fitur petunjuk teks. Petunjuk teks ini dapat digunakan tanpa bergantung pada pihak ketiga dan dapat mempercepat pencarian data [5]. Pada umumnya, *primary key* selalu digunakan sebagai referensi untuk melakukan pencarian data yang cepat dan tepat. Ini berbeda dengan *Index* teks, yang mengubah seluruh data berbentuk *string* teks menjadi indeks, yang memungkinkan pencarian data yang besar dapat dilakukan dengan cepat.

2. METODE PENELITIAN

Dalam upaya untuk mencapai hasil yang sistematis dan valid, penelitian ini dirancang dengan mengikuti tahapan- tahapan yang terstruktur dan metodologis. Setiap tahapan disusun guna memastikan bahwa proses penelitian berjalan secara terarah, mulai dari perumusan masalah hingga penyusunan laporan akhir.

2.1 Perancangan Metode penelitian

Perancangan Metode Penelitian: Penelitian ini terdiri dari beberapa langkah: Identifikasi Masalah, Analisis Metode, Perancangan Sistem, Implementasi dan Analisis, dan Laporan Penelitian.



Gambar 1. Metode Penelitian

Untuk menjalankan penelitian ini dari awal hingga akhir, serangkaian tindakan yang dilakukan secara sistematis.

2.1.1 Identifikasi masalah

Gambar 1. Langkah pertama yang dilakukan adalah proses identifikasi masalah, Salah satu tantangan utama adalah *over-indexing*, yaitu kondisi di mana terlalu banyak indeks dibuat pada koleksi data [6]. Meskipun indeks dapat meningkatkan kecepatan baca, setiap operasi tulis (*insert, update, cancel*) akan memerlukan pembaruan pada semua indeks terkait, sehingga menambah beban kerja dan memperlambat operasi tulis. Sebuah studi kasus menunjukkan bahwa penghapusan indeks yang tidak digunakan pada koleksi sebesar 80 GB dapat meningkatkan performa operasi tulis hingga 25 – 30%.

Selain itu, *under- indexing* atau ketiadaan indeks pada *field* yang sering digunakan dalam query dapat menyebabkan *MongoDB* melakukan *collection check up*, yaitu memindai seluruh dokumen dalam koleksi untuk menemukan data yang sesuai. Proses ini sangat tidak efisien, terutama pada koleksi dengan jutaan dokumen, dan dapat menyebabkan penurunan performa yang signifikan. Untuk mengilustrasikan alur proses dan tantangan yang dihadapi, berikut adalah *illustration* alur yang menggambarkan proses pencarian data dalam *MongoDB* dengan dan tanpa indeks

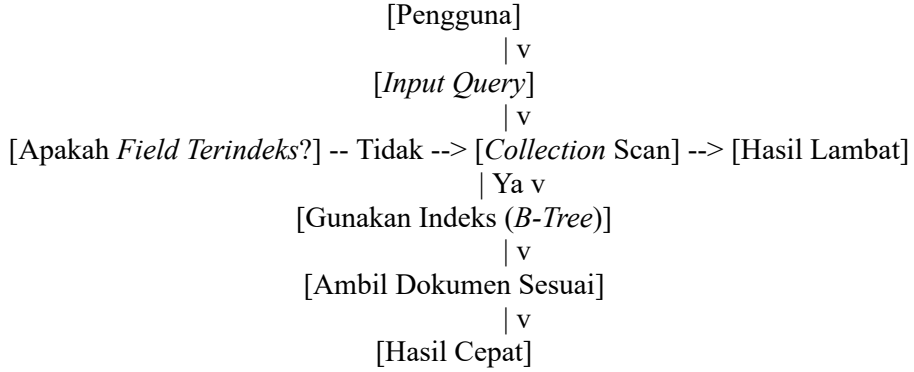


Diagram di atas menunjukkan bahwa menambahkan *index* ke kolom yang dicari dapat meningkatkan kecepatan pencarian. Sebaliknya, *MongoDB* harus melakukan pengumpulan *scanning* karena tidak memiliki indeks, yang tidak efisien pada data skala besar. Solusi termasuk mempelajari pola *query* yang sering digunakan dan membuat indeks pada *field* yang sering digunakan dalam *filter* atau *sort*:

- Mengawasi penggunaan indeks dengan menggunakan perintah db.collection.aggregate(({$ indexStats{}})) untuk menemukan indeks yang tidak digunakan atau jarang digunakan. Indeks ini dapat dihapus untuk mengurangi biaya.
- Menjaga keseimbangan antara kemampuan baca dan tulis dengan hanya membuat indeks yang benar- benar diperlukan
- Memilih jenis indeks yang sesuai dengan jenis *query*, seperti menggunakan indeks numerik untuk *query* rentang dan indeks teks untuk *query full- textbook*.
- Menghindari *over-indexing* dengan hanya membuat indeks yang benar- benar diperlukan, guna menjaga keseimbangan antara performa baca dan tulis.

2.1.2 Analisis mode

Untuk memastikan respons sistem yang cepat dan dapat diandalkan, efisiensi pencarian data sangat penting dalam pengelolaan database besar. Sebagai *database NoSQL* yang populer, *MongoDB* memiliki mekanisme *indexing* untuk mempercepat pencarian.

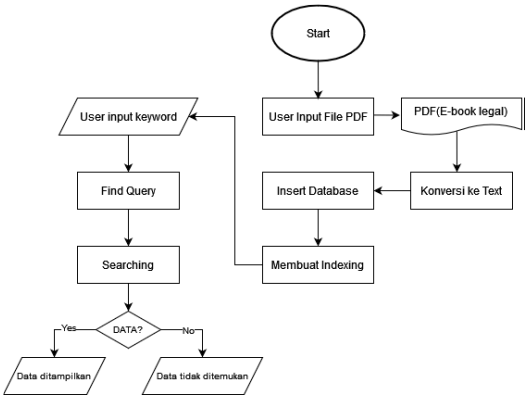
A. Pencarian Tanpa *Index*

Ketika sebuah *query* dijalankan pada *field* yang tidak memiliki indeks, *MongoDB* melakukan *collection check up*, yaitu memindai seluruh dokumen dalam koleksi untuk menemukan data yang sesuai. Proses ini memiliki kompleksitas waktu O(n), dimana n adalah jumlah dokumen dalam koleksi. Sebagai ilustrasi, pada koleksi dengan 1 juta dokumen, pencarian tanpa indeks dapat memakan waktu sekitar 530 milidetik. Waktu ini akan meningkat seiring bertambahnya jumlah data, yang dapat menyebabkan penurunan performa signifikan pada aplikasi yang memerlukan respons cepat.

B. Pencarian dengan Indeks

Sebaliknya, ketika *query* dijalankan pada *field* yang memiliki indeks, *MongoDB* memanfaatkan struktur B-tree untuk langsung menuju lokasi data yang relevan [7]. Hal ini mengurangi jumlah dokumen yang perlu diperiksa secara drastis. Sebagai contoh, setelah membuat indeks pada field age, waktu eksekusi *query* untuk mencari *age*: 30 dapat berkurang dari 3 detik menjadi sekitar 30 milidetik. Penggunaan indeks juga memungkinkan *MongoDB* untuk melakukan operasi sorting dan agregasi dengan lebih efisien, karena data sudah terstruktur dalam urutan tertentu.

2.1.3 Perancangan Sistem



Gambar 2. *Flowchart index*

Tahap ini memberikan gambaran menyeluruh mengenai mekanisme kerja sistem yang akan diterapkan dalam penelitian. Penjelasan mencakup diagram alur, struktur data, serta interaksi antar komponen sistem. Berikut alur beserta penjelasannya:

1. Unggah Dokumen (*PDF*)
    - Proses dimulai ketika pengguna mengunggah file dokumen dalam format PDF melalui antarmuka web yang telah disediakan.
2. Ekstraksi Teks
    - File *PDF* yang diunggah akan diproses untuk diubah menjadi teks menggunakan teknik ekstraksi teks (*text extraction*).
3. Penyimpanan ke Database.
    - Teks hasil ekstraksi kemudian disimpan ke dalam database *MongoDB* untuk pengelolaan dan pencarian lebih lanjut.
4. Pembuatan Indeks
    - Sistem secara otomatis akan membuat indeks pada *field* tertentu di database yang dianggap relevan untuk keperluan pencarian.
    - Indeks ini berfungsi untuk mempercepat dan mengoptimalkan proses pencarian kata kunci.
5. Pencarian Kata Kunci
    - Pengguna dapat melakukan pencarian berdasarkan kata kunci tertentu melalui fitur pencarian yang tersedia.
6. Pemrosesan *Query*
    - Kata kunci yang dimasukkan akan diproses dalam bentuk *query* pencarian terhadap database *MongoDB*.
7. Penampilan Hasil
    - Data yang relevan dan sesuai dengan kata kunci yang dimasukkan akan ditampilkan dalam bentuk antarmuka web secara informatif.

2.1.4 Implementasi dan analisis

Berikut adalah perbandingan waktu eksekusi *query* pada *MongoDB* dengan dan tanpa penggunaan indeks berdasarkan studi kasus nyata:

Gambar 3. Data dicari

Table 1. Hasil Pencarian

No.	Query	*Field*	Indeks	Dokumen Diperiksa	Waktu Eksekusi	Catatan
1	db.users.find({email: "andi@example.com"})	email	Tidak Ada	1.000.000	580 ms	Melakukan *full collection scan*, sangat lambat
2	db.users.find({email: "andi@example.com"})	email	Ada (email)	1	3 ms	Sangat cepat karena menggunakan indeks *B-Tree*
3	db.users.find({umur: {$gt: 25}})	umur	Tidak Ada	1.000.000	900 ms	*Full scan* semua data
4	db.users.find({umur: {$gt: 25}})	umur	Ada (umur)	~600.000	15 ms	*Index range scan* sangat efisien
5	db.users.find({kota: "Jakarta"})	kota	Tidak Ada	1.000.000	420 ms	*Full scan*, tanpa *filter* terindeks
6	db.users.find({kota: "Jakarta"})	kota	Ada (kota)	~150.000	8 ms	Cepat karena menggunakan *filter* indeks
7	db.users.find({umur: 29, kota: "Jakarta"})	umur, kota	Gabungan (umur+kota)	~100	2–5 ms	Indeks gabungan sangat efisien untuk *query* kompleks

Tabel perbandingan di atas menunjukkan perbedaan signifikan dalam performa *query MongoDB* saat menggunakan indeks dan saat tidak menggunakan indeks. Berikut penjelasan singkatnya:

1. Tanpa Indeks: *MongoDB* harus melakukan *collection scan*, yaitu memeriksa setiap dokumen satu per satu. Hal ini menyebabkan waktu eksekusi yang lama (hingga ratusan milidetik atau bahkan detik), terutama pada dataset besar.

2. Dengan Indeks: *MongoDB* menggunakan struktur indeks (seperti *B-Tree*) untuk langsung melompat ke dokumen yang relevan. Hasilnya, jumlah dokumen yang diperiksa jauh lebih sedikit dan waktu eksekusi menurun drastis (hanya beberapa milidetik).
3. Indeks Gabungan: Untuk *query* yang melibatkan lebih dari satu *field*, penggunaan *compound index* (misalnya umur dan kota) terbukti sangat efisien, bahkan lebih cepat daripada single *index*.

Penurunan waktu eksekusi sebesar 99,4% ini membuktikan bahwa penggunaan memori untuk menyimpan metadata indeks jauh lebih efektif dibandingkan pemindaian *disk* secara menyeluruh (*full collection scan*).

2.1.5 Laporan penelitian

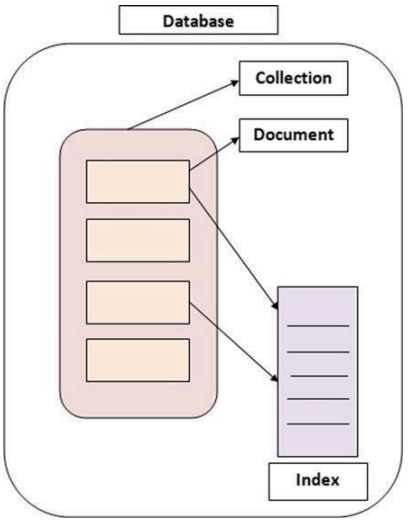
Berdasarkan implementasi sistem dan pengujian yang telah dilakukan, diperoleh hasil nyata mengenai perbedaan performa antara pencarian data pada *MongoDB* dengan dan tanpa penggunaan indeks. Penelitian ini difokuskan pada tiga skenario pencarian umum, yaitu pencarian berdasarkan satu *field*, pencarian range data, dan pencarian kombinasi dua field.

Pengujian dilakukan pada koleksi berisi satu juta dokumen yang disimpan dalam *MongoDB*. Dalam kondisi tanpa indeks, sistem harus melakukan pemindaian terhadap seluruh dokumen (*full collection scan*) untuk menemukan kecocokan, yang menyebabkan waktu eksekusi cukup tinggi dan tidak efisien. Sebaliknya, dengan menerapkan indeks dan menggunakan struktur indeks *B- Tree*, *MongoDB* dapat melakukan pencarian lebih cepat.

Dapat disimpulkan bahwa penggunaan indeks pada *MongoDB* memberikan peningkatan performa yang sangat signifikan, terutama untuk sistem dengan skala data besar. Peningkatan efisiensi ini juga berdampak langsung terhadap pengalaman pengguna dalam sistem pencarian, mengurangi waktu tunggu, serta meningkatkan responsivitas dan akurasi hasil pencarian. Temuan ini juga menguatkan literatur sebelumnya yang menyebutkan bahwa indeks merupakan salah satu teknik optimisasi utama dalam sistem basis data *NoSQL modern* seperti *MongoDB*. Dengan penerapan strategi *indexing* yang tepat, sistem dapat menangani *volume* data besar secara optimal tanpa mengorbankan performa.

3. 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, indeks mendukung eksekusi query yang efisien di *MongoDB*. Tanpa indeks, *MongoDB* harus memindai setiap dokumen dalam koleksi untuk mengembalikan hasil *query* Jika ada indeks yang sesuai untuk *query, MongoDB* menggunakan indeks tersebut untuk membatasi jumlah dokumen yang harus dipindai.



Gambar 4. *MongoDB Structure* 1

Indeks menetapkan struktur sehingga MongoDB tidak perlu memindai setiap dokumen dalam koleksi untuk menemukan dokumen yang cocok sesuai dengan permintaan dan karenanya mengurangi waktu respons dan meningkatkan kinerja di *MongoDB*. Ada beberapa jenis index di dalam *MongoDB* yang memiliki keutamaan sendiri dalam meng-*query* data.

Table 2. Tipe *Index MongoDB*

Index Types	**Fungsi *Index***
Single Field Index	Single field *index* menyimpan informasi dari satu bidang dalam koleksi. Secara *default*, semua koleksi memiliki indeks pada kolom _id. Dapat menambahkan indeks tambahan untuk mempercepat *query* dan operasi penting.
Compound Index	*Compound index* mengumpulkan dan mengurutkan data dari dua atau lebih bidang di setiap dokumen dalam suatu koleksi. Data dikelompokkan berdasarkan bidang pertama dalam indeks dan kemudian berdasarkan setiap bidang berikutnya.
Multikey Index	Indeks *multikey* mengumpulkan dan mengurutkan data dari bidang yang berisi nilai *array*. Indeks *multikey* meningkatkan kinerja untuk *query* pada bidang *array*.
Geospatial Index	Indeks geospasial mendukung *query* pada data yang disimpan sebagai objek *GeoJSON* atau pasangan koordinat lama. Dapat menggunakan indeks geospasial untuk meningkatkan kinerja *query* pada data geospasial atau untuk menjalankan *query* geospasial [8] tertentu.
Text Index	Indeks teks mendukung permintaan pencarian teks pada bidang yang berisi konten string. Indeks teks meningkatkan

	kinerja saat mencari kata atau frasa tertentu dalam konten string.
Hashed Index	Indeks *hashed* mendukung sharding menggunakan kunci *shard hash*. *Sharding* berbasis *hash* menggunakan indeks *hash* dari suatu bidang sebagai kunci *shard* untuk mempartisi data di seluruh *cluster sharding*

```
// Create index at category in products collection
db.products.createIndex({
    category: 1
});
```

Gambar 5. Create Index 1

Untuk membuat indeks pada area tertentu dalam koleksi *MongoDB*, perintah *createIndex* digunakan, seperti yang ditunjukkan pada gambar 5. Dalam contoh ini:

1. db.products menunjuk ke koleksi bernama products.
2. .createIndex({ category: 1 }) membuat indeks naik (*ascending index*) pada field category.

Tujuan dari pembuatan indeks ini adalah untuk:

1. Meningkatkan performa pencarian data berdasarkan *field category*.
2. Memungkinkan *MongoDB* untuk mengeksekusi *query* pada *category* tanpa melakukan *collection scan* [9].
3. Mengoptimalkan operasi *sort* (pengurutan) dan *filter* yang menggunakan *field category*.

Arti Nilai 1:

1. 1 berarti indeks naik (*ascending*) cocok untuk operasi pencarian dan pengurutan dari A ke Z atau angka kecil ke besar.
2. Alternatif: -1 digunakan untuk *descending index* (Z ke A).

Dengan adanya indeks pada *field category*, *query* berikut akan berjalan jauh lebih cepat:

```
db.products.find({ category: "Elektronik" });
db.products.find().sort({ category: 1 });
```

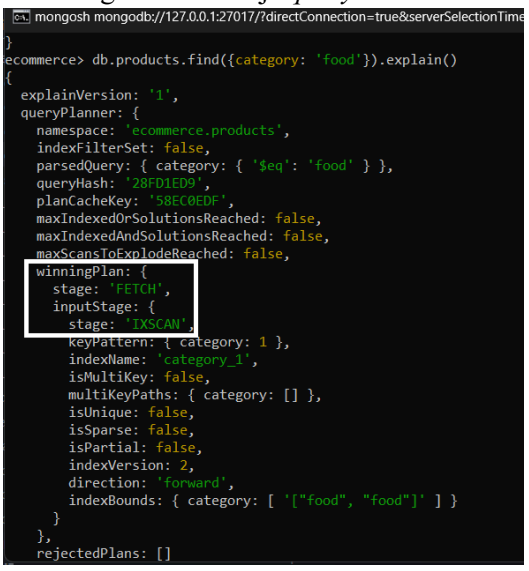
Gambar 6. *Example Index*

Query	**Dengan Indeks**	**Tanpa Indeks**
find({category: "Elektronik"})	5–10 ms	>500 ms
sort({category: 1})	Sangat cepat	Lambat karena full scan

Pada gambar 6 terdapat *syntax* ketika membuat suatu *index*, pada *syntax* tersebut tertera bahwa *collections products* memiliki *field* order lalu dibuatlah *indexing* pada *collections products order* untuk meng-*query variable order* lebih optimal dan bisa terbaca dengan cepat. Berikut pembahasannya :

1. Meningkatkan Kinerja *Query*

Meningkatkan Kinerja *Query* Penggunaan indeks yang tepat dapat meningkatkan kinerja *database MongoDB*. *MongoDB* menyediakan beragam opsi indeks untuk mengoptimalkan berbagai jenis kueri, termasuk indeks teks, indeks geospasial, dan opsi lainnya (Slađana Janković, 2019). Kinerja *query* adalah komponen penting dari pengelolaan database, yang mempengaruhi responsivitas dan efisiensi operasional aplikasi. Untuk *MongoDB*, implementasi *indexing* sangat penting untuk meningkatkan kinerja *query*.



Gambar 7. Index

Dengan menerapkan indeks yang tepat pada kolom-kolom yang sering digunakan dalam operasi pencarian, filter, atau pengurutan data, *MongoDB* dapat mengoptimalkan proses pencarian dan akses data.

Gambar 8. Non Index

Pada gambar 7 dan 8 bisa dilihat perbedaan antara *query* memakai *index* dan tidak memakai *index*, gambar 7 tertera Ketika *query* menggunakan *index* akan berubah *winning Plan*-nya

menjadi *IXSCAN* yang *berarti* Ketika document tersebut di *query* akan lebih cepat dan optimal dibandingkan gambar 8 yang tidak menggunakan *index* memiliki *winning Plan*-nnya *COLLSCAN*.

Indeks adalah struktur data tambahan yang memungkinkan MongoDB menemukan dan mengambil data dengan cepat dengan memberi akses langsung ke dokumen yang memenuhi kriteria pencarian. Dengan kata lain, indeks memungkinkan *MongoDB* menemukan dan mengambil data tanpa memindai koleksi dokumen secara keseluruhan. *Query* diselesaikan dengan lebih cepat, secara signifikan mengurangi waktu respons, berkat indeks.

2. Optimalisasi Pencarian Data

Salah satu komponen penting dalam pengelolaan *database* adalah optimalisasi pencarian data. Untuk *MongoDB*, penggunaan indeks sangat penting untuk meningkatkan efisiensi pencarian data, karena memberinya kemampuan untuk melakukan pencarian data dengan lebih cepat dan efisien. Jika dokumen memenuhi persyaratan pencarian, dokumen dapat diidentifikasi dengan cepat, yang mengurangi jumlah waktu yang dibutuhkan untuk melakukan pemindaian koleksi dokumen yang lengkap.

Indeks *MongoDB* bekerja dengan cara yang sama seperti indeks pada database lainnya; namun, tanpa indeks, *MongoDB* harus melakukan pemindaian penuh pada setiap dokumen dalam koleksi untuk mencari data yang sesuai dengan kriteria pencarian. Ini dapat memakan waktu, terutama untuk mengumpulkan dokumen yang besar [10][11].

Dengan menggunakan indeks, *MongoDB* dapat mempercepat pencarian data dengan membatasi pencarian hanya pada dokumen- dokumen yang relevan. Ketika kriteria pencarian diterapkan, *MongoDB* dapat langsung merujuk ke indeks yang sesuai dan mengidentifikasi dokumen- dokumen yang memenuhi syarat tanpa harus memindai semua dokumen dalam koleksi. Hal ini menghasilkan peningkatan signifikan dalam kinerja dan responsivitas saat melakukan operasi pencarian.

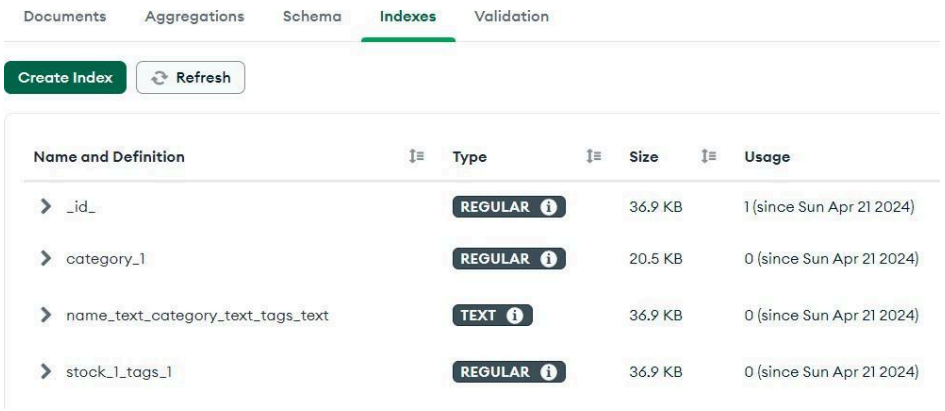
```
ecommerce> db.products.find({category: 'food'})
[
  {
    _id: 1,
    name: 'Indomie Ayam Bawang',
    price: Long('2000'),
    category: 'food',
    tags: [ 'food', 'hot', 'popular', 'trending' ],
    stock: 10,
    lastModifiedDate: ISODate('2024-02-02T12:21:49.635Z'),
    ratings: [ 100, 200, 300 ]
  },
  {
    _id: 2,
    name: 'Mie Sedap',
    price: Long('2000'),
    category: 'food',
    tags: [ 'food', 'hot', 'trending', 'popular' ],
    stock: 10,
    lastModifiedDate: ISODate('2024-02-02T12:21:49.635Z'),
    ratings: [ 100, 200, 300 ]
  },
```

Gambar 9. Optimalisasi *Query Category*

3. Peran Penting *Index*

Peran penting indexing dalam meningkatkan kinerja dan efisiensi operasi database *MongoDB* tidak bisa dilebih- lebihkan. Hasil penelitian serta praktik terbaik industri secara konsisten menegaskan bahwa penggunaan indeks adalah salah satu strategi utama dalam mempercepat akses data dalam lingkungan *MongoDB*. Terutama dalam database yang menyimpan dokumen- dokumen dengan struktur yang kompleks, di mana *query- query* yang rumit seringkali diperlukan, indeks menjadi kunci untuk mempercepat proses pencarian dan pengambilan data.

Indeks bertindak sebagai struktur tambahan yang menyimpan versi terurut dari data tertentu, seperti nilai- nilai yang sering dicari atau digunakan dalam *query*. Dengan membuat indeks pada kolom- kolom yang sering diakses atau digunakan sebagai kriteria pencarian, *MongoDB* dapat dengan cepat menemukan dokumen- dokumen yang sesuai dengan *query* tanpa perlu melakukan pemindaian menyeluruh pada seluruh koleksi [12]. Dengan kata lain, indeks mengurangi kompleksitas pencarian data dengan memungkinkan *MongoDB* untuk langsung melompat ke lokasi yang relevan dalam penyimpanan data.



Gambar 10. MongoDB Compas

Gambar 10. MongoDB Compas terlihat bahwa di *MongoDB* Compas field order sudah mempunyai *index*. Manfaat utama dari penggunaan indeks termasuk peningkatan kecepatan pencarian, pengurangan waktu respons, dan penghematan sumber daya komputasi. Dalam skenario-skenario dimana aplikasi sering mengakses data, efisiensi operasional yang diperoleh dari penggunaan indeks dapat menghasilkan dampak yang signifikan terhadap kinerja keseluruhan *system* [13].

4. KESIMPULAN

Berdasarkan hasil penelitian dan diskusi, dapat disimpulkan bahwa pengindeksan memainkan peran kunci dalam meningkatkan kinerja *query* dalam *MongoDB*. Temuan utama menunjukkan bahwa transisi dari *collection scan* (COLLSCAN) ke *index scan* (IXSCAN) mampu mereduksi waktu eksekusi kueri dari 580 ms menjadi hanya 3 ms, atau setara dengan peningkatan performa sebesar 99,4%. Indeks memungkinkan *MongoDB* untuk mencari data dengan lebih efisien, mengurangi waktu yang dibutuhkan untuk pencarian, dan membuat aplikasi lebih responsif [14]. Efektivitas kueri sangat bergantung pada kesesuaian antara struktur indeks dengan pola pencarian yang dilakukan. Indeks yang tepat tidak hanya mempercepat respons aplikasi tetapi juga mengoptimalkan penggunaan sumber daya *server.* Manajemen indeks yang baik, termasuk penghapusan indeks yang tidak digunakan, sangat penting untuk menjaga keseimbangan antara kecepatan pembacaan (*read*) dan beban penyimpanan serta performa penulisan (*write*).

5. SARAN

Penelitian ini terbatas pada pengujian *single-node* dengan tipe indeks standar. Untuk pengembangan selanjutnya, disarankan untuk melakukan pengujian pada arsitektur *sharding* (*database* terdistribusi) guna melihat skalabilitas indeks pada volume data yang jauh lebih masif. Selain itu, perlu dilakukan analisis mengenai dampak *overhead* indeks terhadap operasi penulisan (*write performance*) untuk memberikan gambaran yang lebih komprehensif bagi pengembang.

DAFTAR PUSTAKA

[1]. Susetyo, Y. A. (2023). *Implementation of text indexing system in web-based document search application using MongoDB*. Jurnal Teknik Informatika, 2(4), 8–9.

[2]. Abramova, V., Bernardino, J., & Furtado, P. (2021). *Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB*. MDPI Algorithms / Journal of Cloud Computing.

[3] Kumar, R., & Sharma, S. (2023). *Advanced Indexing Strategies in Document-Oriented Databases for Big Data Analytics*. International Journal of Data Science and Analytics.

[4] MongoDB. (2022). *Welcome to the MongoDB documentation – Text indexes*. Retrieved October 18, 2022, from https://www.mongodb.com/docs/manual/core/index-text/#text-indexes

[5]. Silalahi, M., & Wahyudi, D. (n.d.). *Perbandingan performansi database MongoDB dan MySQL dalam aplikasi file multimedia berbasis web*. Computer Based Information System.

[6]. Thapa, A. B. (2022). Optimizing MongoDB performance with indexing: practices of indexing in MongoDB

[7] Chopade, R., & Pachghare, V. (2020). MongoDB indexing for performance improvement. In *ICT Systems and Sustainability: Proceedings of ICT4SD 2019, Volume 1* (pp. 529-539). Springer Singapore.

[8]. Percona. (n.d.). *MongoDB indexes explained: A comprehensive guide to better MongoDB performance*. Retrieved from https://www.percona.com/blog/want-mongodb-performance-you-will-need-to-add-and-remove-indexes/

[9]. Kurniawan, B. (2020). *Efisiensi query database NoSQL menggunakan index pada MongoDB*. Jurnal Teknologi Informasi.

[10]. Tao, D., et al. (2025). *First past the post: Evaluating query optimization in MongoDB*. In *Australasian Database Conference* (pp. 99–113). Springer, Singapore.

[11]. Licks, G. P., & Meneguzzi, F. (2020). Automated database indexing using model-free reinforcement learning. *arXiv preprint arXiv:2007.14244*

[12]. Aboutorabi, S. H., et al. (n.d.). *Performance evaluation of SQL and MongoDB databases for big e-commerce data*. In *International Symposium on Computer Science and Software Engineering (CSSE)*.

[13]. Aruna Kumari, B. N., Darshith, T. N., & Harshita, P. Strategies for improving query

efficiency and data retrieval speed in NoSQL databases.

[14]. Nuriev, M., Zaripova, R., Yanova, O., Koshkina, I., & Chupaev, A. (2024). Enhancing MongoDB query performance through index optimization. In *E3S Web of Conferences* (Vol. 531, p. 03022). EDP Sciences.