OPTIMASI QUERY SQL SERVER DENGAN TEKNIK INDEXING DAN PERFORMANCE MONITORING

Gloria Raphaela Mei Lanny Br Aritonang, Mhd. Arief Hasan*, Mohammad khasbulla Ridwan, Muhamad Nur Iman, Ricky Carlo Pratama Silalahi, Rizky Suranta Sipayung
Teknik Informatika, Universitas Lancang Kuning
Jl. Yos Sudarso No.KM. 8, Umban Sari, Kec. Rumbai, Kota Pekanbaru, Riau 28266
m.arif@unilak.ac.id

ABSTRAK

Pengelolaan database yang efisien merupakan salah satu tantangan utama dalam dunia teknologi informasi, terutama pada sistem dengan volume data besar dan kompleksitas query yang tinggi. Penelitian ini bertujuan untuk mengoptimalkan kinerja query pada Microsoft SQL Server melalui penerapan teknik indexing dan monitoring performa. Eksperimen dilakukan dengan membandingkan performa query pada database tanpa indeks dan dengan indeks, menggunakan data dummy sebanyak 1 juta baris. Hasil penelitian menunjukkan bahwa penerapan indeks mampu mengurangi logical reads hingga **89.77%**, CPU time hingga **68.09%**, dan waktu eksekusi (elapsed time) hingga **90.20%**. Selain itu, monitoring performa melalui SQL Server Management Studio (SSMS) mengidentifikasi tingkat fragmentasi indeks sebagai faktor yang memengaruhi efisiensi query. Dengan demikian, teknik indexing terbukti efektif dalam meningkatkan kinerja pengolahan data pada database SQL Server. Penelitian ini memberikan wawasan strategis tentang pentingnya perancangan indeks yang tepat serta perlunya monitoring performa untuk menjaga efisiensi sistem database.

Kata kunci: Optimasi Query, SQL Server, Indexing, Monitoring Performa, Efisiensi Database

1. PENDAHULUAN

Pertumbuhan data yang pesat di era digital saat ini telah menghadirkan tantangan baru bagi pengelolaan dan pengolahan data dalam sistem basis data. Berbagai organisasi, baik skala kecil maupun besar, bergantung pada performa sistem manajemen basis data untuk mendukung operasional dan pengambilan keputusan strategis [1]. Dalam hal ini, efisiensi eksekusi query menjadi salah satu faktor utama yang menentukan keberhasilan pengelolaan data. Namun, sering kali sistem mengalami penurunan performa seiring bertambahnya volume data, sehingga mengakibatkan waktu respon yang lebih lama dan penggunaan sumber daya yang tidak optimal.

SQL Server, sebagai salah satu sistem manajemen basis data relasional terkemuka, menyediakan berbagai fitur untuk meningkatkan performa query [2]. Salah satu teknik yang paling efektif adalah penggunaan indexing. Indeks merupakan struktur data khusus yang dirancang untuk mempercepat proses pencarian dan pengambilan data. Dengan indeks, sistem dapat mengurangi jumlah operasi baca yang diperlukan untuk mengeksekusi query, sehingga menghasilkan waktu eksekusi yang lebih cepat. Namun, implementasi indeks yang tidak tepat dapat menyebabkan overhead, seperti meningkatnya waktu pembaruan data dan penggunaan ruang penyimpanan.

Selain indexing, teknik monitoring performa juga menjadi aspek krusial dalam memastikan efisiensi sistem. SQL Server menyediakan alat seperti SQL Server Profiler dan Dynamic Management Views (DMVs) untuk memantau aktivitas query dan mengevaluasi penggunaan sumber daya. Monitoring ini memungkinkan administrator basis data untuk mengidentifikasi bottleneck, seperti query yang tidak efisien atau fragmentasi indeks, dan melakukan tindakan korektif yang diperlukan [3].

Dalam penelitian ini, kami mengeksplorasi optimasi query pada SQL Server menggunakan kombinasi teknik indexing dan performance monitoring. Studi ini berfokus pada bagaimana indeks dapat dirancang dan diterapkan untuk mendukung query yang kompleks serta bagaimana monitoring dapat digunakan untuk memastikan bahwa sistem berjalan pada kondisi optimal [4]. Dengan memanfaatkan kedua teknik tersebut, diharapkan performa sistem dapat ditingkatkan secara signifikan tanpa mengorbankan konsistensi dan integritas data.

Penelitian ini bertujuan untuk memberikan panduan praktis bagi praktisi teknologi informasi dan administrator basis data dalam mengoptimalkan query SQL Server. Dengan studi kasus yang melibatkan pembuatan data skala besar dan pengujian performa query sebelum dan sesudah implementasi indeks, kami menunjukkan pentingnya perencanaan indeks yang tepat dan pemantauan sistem secara berkala. Hasil dari penelitian ini diharapkan dapat menjadi acuan dalam mengelola sistem basis data yang efisien dan handal, terutama dalam menghadapi tantangan pengelolaan data yang terus berkembang.

2. TINJAUAN PUSTAKA

Tinjauan ini bertujuan untuk memberikan pemahaman mendalam mengenai konteks studi serta mendukung penyusunan kerangka berpikir yang digunakan dalam penelitian ini.

Penelitian terdahulu yang dilakukan oleh[5], dengan judul "Perbandingan Performa MySQL dan MongoDB pada Sistem Informasi Pengelolaan Air

Limbah di Bali" mendapatkan hasil dari pengujian mengindikasikan bahwa MongoDB memiliki keunggulan yang jelas dibandingkan MySQL di semua variabel yang diuji. Waktu respon MongoDB lebih cepat hingga 49.35%, disertai peningkatan throughput rata-rata mencapai 54.5%. Pada saat beban maksimal, MongoDB mencatat tingkat kesalahan sebesar 0.67% sementara MySQL menunjukkan 1.23%, yang mencerminkan stabilitas yang lebih baik. Pemanfaatan sumber daya juga lebih efisien dengan penggunaan CPU yang 15.7% lebih rendah dibandingkan dengan MySQL. Temuan ini sesuai dengan tujuan penelitian awal yang bertujuan menemukan solusi terbaik untuk menghadapi tantangan performa sistem DSDP. MongoDB terbukti lebih efisien dalam mengelola kueri yang kompleks serta volume data yang besar tanpa memerlukan operasi penggabungan, yang menjadi salah satu kendala utama dalam implementasi MySQL saat ini. Berdasarkan hasil tersebut, disarankan untuk melakukan migrasi ke MongoDB sebagai cara untuk meningkatkan performa sistem DSDP. Implementasi sebaiknya dilakukan secara bertahap dengan memperhatikan aspek optimasi sistem, pemantauan, dan pelatihan tim.

2.1. Database

Database adalah kumpulan data yang terorganisasi dan disimpan secara sistematis untuk mendukung pengelolaan dan akses data secara efisien. Dalam sistem modern, database digunakan untuk menyimpan informasi dalam jumlah besar dengan tetap menjaga integritas dan konsistensi data. Sistem Manajemen Basis Data (Database Management System atau DBMS) berperan penting dalam mengelola database. Contoh DBMS yang banyak digunakan meliputi MySQL, PostgreSQL, dan Microsoft SQL Server. Dalam dunia bisnis, database mempermudah pelacakan transaksi, manajemen inventaris, dan analisis data untuk pengambilan Keputusan [6].

2.2. Query

Query adalah perintah atau instruksi yang digunakan untuk mengakses dan memanipulasi data dalam database. Dengan menggunakan Structured Query Language (SQL), pengguna dapat menjalankan berbagai operasi, seperti pencarian data (SELECT), pembaruan data (UPDATE), penghapusan data (DELETE), dan penambahan data (INSERT). Efisiensi query sangat penting dalam sistem database untuk memastikan bahwa waktu respons tetap cepat, terutama saat menangani volume data yang besar. Teknik optimasi, seperti indexing, dapat secara signifikan meningkatkan performa query [7].

2.3. SQL Server

SQL Server adalah sistem manajemen basis data relasional (RDBMS) yang dikembangkan oleh Microsoft. SQL Server menyediakan berbagai fitur unggulan, termasuk dukungan untuk indexing, teknik optimasi query, dan alat monitoring performa. Sistem ini dirancang untuk menangani berbagai kebutuhan pengelolaan data, mulai dari transaksi operasional hingga analitik data skala besar. Dengan integrasi ke platform seperti Azure dan Power BI, SQL Server menjadi pilihan utama dalam pengelolaan data untuk berbagai sektor industry [8].

2.4. Indexing

Indexing adalah teknik optimasi database yang bertujuan untuk meningkatkan kecepatan eksekusi query. Indeks bekerja dengan cara menciptakan struktur data tambahan yang mempermudah pencarian data. Dalam SQL Server, terdapat dua jenis indeks utama: clustered index dan non-clustered index. Clustered index menentukan urutan fisik data dalam tabel, sedangkan non-clustered index menciptakan pointer ke data. Dengan penerapan indeks yang tepat, waktu eksekusi query dapat berkurang secara signifikan, terutama untuk query yang melibatkan pencarian atau pengurutan data [9].

2.5. Performance Monitoring

Performance monitoring adalah proses memantau dan menganalisis kinerja sistem database untuk mengidentifikasi bottleneck atau masalah lainnya. Dalam SQL Server, alat seperti Dynamic Management Views (DMVs) dan SQL Server Profiler digunakan untuk memantau aktivitas query dan penggunaan sumber daya. Monitoring ini memungkinkan administrator database untuk mendeteksi query yang lambat, fragmentasi indeks, atau penggunaan CPU yang berlebihan. Dengan pemantauan yang konsisten, sistem dapat dioptimalkan untuk mencapai kinerja yang lebih baik.

2.6. Dynamic Management Views (DMVs)

DMVs adalah fitur bawaan SQL Server yang menyediakan informasi mendalam tentang status dan aktivitas database. DMVs membantu dalam memantau penggunaan indeks, statistik query, dan alokasi memori. Misalnya, sys.dm_db_index_physical_stats digunakan untuk mengevaluasi tingkat fragmentasi indeks, sedangkan sys.dm_exec_query_stats menyediakan informasi tentang performa Penggunaan DMVs sangat penting dalam strategi optimasi dan troubleshooting performa SQL Server [10].

2.7. Fragmentasi Indeks

Fragmentasi indeks terjadi ketika urutan logis data dalam indeks tidak sesuai dengan urutan fisiknya, yang dapat memperlambat waktu akses data. Fragmentasi sering terjadi akibat pembaruan, penghapusan, atau penambahan data dalam tabel. SQL Server menyediakan alat untuk memeriksa tingkat fragmentasi, seperti sys.dm_db_index_physical_stats, dan mendukung tindakan seperti reorganisasi atau

rekonstruksi indeks untuk mengurangi fragmentasi dan meningkatkan performa query.

2.8. Execution Plan

Execution plan adalah representasi grafis atau tekstual dari langkah-langkah yang diambil SQL Server untuk mengeksekusi query. Dengan melihat execution plan, pengguna dapat memahami bagaimana query diolah, apakah indeks dimanfaatkan dengan baik, dan di mana potensi bottleneck berada. Analisis execution plan membantu dalam mengidentifikasi query yang tidak efisien dan menentukan strategi optimasi yang sesuai, seperti menambahkan indeks atau memodifikasi struktur query.

2.9. SQL Server Management Studio (SSMS)

SSMS adalah alat resmi yang dikembangkan oleh Microsoft untuk mengelola SQL Server. Dengan antarmuka grafis yang intuitif, SSMS memungkinkan pengguna untuk membuat tabel, menulis query, memantau performa sistem, dan menganalisis execution plan. SSMS juga mendukung fitur indexing dan monitoring performa, menjadikannya alat yang sangat penting bagi administrator dan pengembang database dalam mengelola sistem yang kompleks [11].

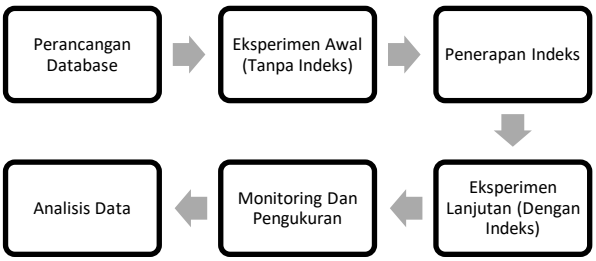
2.10. Optimasi Query

Optimasi query adalah proses yang bertujuan untuk meningkatkan efisiensi eksekusi query dengan meminimalkan waktu proses dan penggunaan sumber daya. Teknik ini melibatkan penggunaan indeks, penghapusan sub-query yang tidak perlu, dan penyusunan ulang query untuk memanfaatkan execution plan yang lebih optimal. Dalam sistem database besar, optimasi query merupakan kunci untuk memastikan bahwa sistem dapat menangani permintaan data secara cepat dan konsisten [12].

3. METODE PENELITIAN

3.1. Tahapan Penelitian

Tahapan penelitian adalah seperti berikut.



Gambar 2. Tahapan penelitian

3.2. Perancangan Database

Tahap awal penelitian ini adalah merancang struktur database yang digunakan untuk eksperimen. Database dirancang dengan tabel utama bernama *Produk* yang memiliki beberapa kolom, seperti ID_Produk, Nama_Produk, Kategori, Harga, dan Stok. Data dummy sebanyak 1 juta record dihasilkan secara otomatis dengan variasi pada kolom Kategori dan Harga untuk mencerminkan kondisi dunia nyata. Database ini diimplementasikan pada SQL Server menggunakan SQL Server Management Studio (SSMS). Pembuatan database dapat dilihat pada Gambar 2.

```
CREATE DATABASE OptimasiQueryDB;
GO
USE OptimasiQueryDB;
GO
CREATE TABLE Produk (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    NamaProduk NVARCHAR(100),
    Kategori NVARCHAR(50),
    Harga DECIMAL(10,2),
    Stok INT,
    TanggalDitambahkan DATE DEFAULT GETDATE()
);
SET NOCOUNT ON;
DECLARE @batchSize INT = 10000;
DECLARE @TotalRows INT = 1000000;
DECLARE @i INT = 1;
WHILE @i <= @TotalRows
BEGIN
    INSERT INTO Produk (NamaProduk, Kategori, Harga, Stok)
    SELECT
        CONCAT('Produk_', FORMAT(@i + s.RowNumber - 1, '000000')) AS NamaProduk,
        CASE
            WHEN (s.RowNumber + @i - 1) % 3 = 0 THEN 'Elektronik'
            WHEN (s.RowNumber + @i - 1) % 3 = 1 THEN 'Furnitur'
            ELSE 'Pakaian'
        END AS Kategori,
        ROUND(RAND(CHECKSUM(NEWID())) * 1000000, 2) AS Harga,
        FLOOR(RAND(CHECKSUM(NEWID())) * 100) + 1 AS Stok
    FROM (SELECT TOP (@batchSize) ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNumber FROM sys.all_objects) s;

    SET @i = @i + @batchSize;
END
```

Gambar 2. Pembuatan database

3.3. Eksperimen Awal (Tanpa Indeks)

Pada eksperimen awal, pengujian dilakukan tanpa menggunakan indeks untuk mengukur baseline performa query. Query yang digunakan memiliki tingkat kompleksitas sedang hingga tinggi, seperti pencarian data berdasarkan kolom *Kategori* dan penghitungan agregat. Data performa, seperti waktu eksekusi query, logical reads, dan CPU time, dicatat menggunakan fitur SET STATISTICS IO dan SET STATISTICS TIME untuk memberikan gambaran kondisi awal.

3.4. Penerapan Indeks

Tahap selanjutnya adalah penerapan indeks untuk meningkatkan performa query. Teknik indexing yang digunakan meliputi clustered index pada kolom ID_Produk sebagai primary key dan non-clustered index pada kolom Kategori untuk mempercepat pencarian data. Proses ini mencakup perancangan indeks, implementasi melalui perintah SQL, dan verifikasi untuk memastikan indeks telah dibuat dan berfungsi dengan baik. Pembuatan indeks dapat dilihat pada Gambar 3.

```
CREATE INDEX idx_Kategori_Harga_Stok ON Produk (Kategori, Harga, Stok);
GO
```

```
Messages
Commands completed successfully.
Completion time: 2025-02-08T16:48:55.5744944+07:00
```

Gambar 3. Pembuatan indeks

3.5. Eksperimen Lanjutan (Dengan Indeks)

Setelah indeks diterapkan, eksperimen lanjutan dilakukan untuk mengukur peningkatan performa. Query yang sama dengan eksperimen awal dijalankan kembali, dan metrik performa dicatat untuk membandingkan hasil dengan baseline sebelumnya. Data seperti waktu eksekusi query, logical reads, dan fragmentasi indeks dianalisis untuk menentukan efektivitas penerapan indeks.

3.6. Monitoring dan Pengukuran

Monitoring dilakukan untuk mengevaluasi keberlanjutan kinerja indeks yang telah diterapkan. Alat bawaan SQL Server, seperti Dynamic Management Views (DMVs), digunakan untuk memantau tingkat fragmentasi indeks dan penggunaan sumber daya. Data monitoring memberikan informasi penting tentang kesehatan indeks, seperti fragmentasi tinggi yang dapat memengaruhi performa query.

3.7. Analisis Data

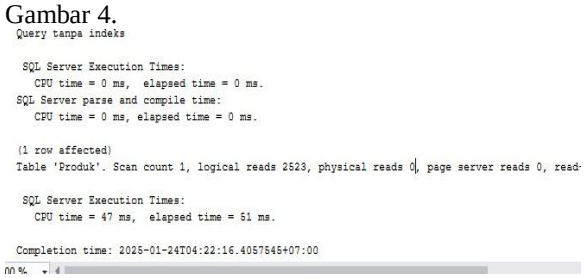
Tahap akhir adalah analisis data untuk mengevaluasi dampak penerapan indeks terhadap performa query. Hasil eksperimen awal dan lanjutan dibandingkan untuk menghitung peningkatan efisiensi, baik dari sisi waktu eksekusi maupun penggunaan sumber daya. Berdasarkan hasil analisis, rekomendasi diberikan terkait penerapan teknik indexing pada skenario serupa, serta strategi pemeliharaan indeks untuk menjaga performa sistem tetap optimal.

4. HASIL DAN PEMBAHASAN

Pada bagian ini akan dijelaskan hasil eksperimen yang telah dilakukan untuk mengukur performa query sebelum dan sesudah penerapan indeks. Pembahasan meliputi analisis performa query, dampak penerapan indeks terhadap kecepatan eksekusi, serta evaluasi monitoring yang dilakukan.

4.1. Eksperimen Awal (Tanpa Indeks)

Hasil Pada tahap awal, query dijalankan tanpa menggunakan indeks pada tabel *Produk*. Hasil eksperimen menunjukkan bahwa query membutuhkan waktu eksekusi yang cukup lama, terutama saat jumlah record dalam tabel mencapai 1 juta, bisa dilihat pada Gambar 4.



Gambar 4. Hasil tanpa indeks

Berikut adalah metrik performa yang dicatat dapat dilihat pada Tabel 1.

Tabel 1. Tabel performa awal

Parameter	Nilai
Logical Reads	2,512 pages
CPU Time	47 ms
Elapsed Time	51 ms

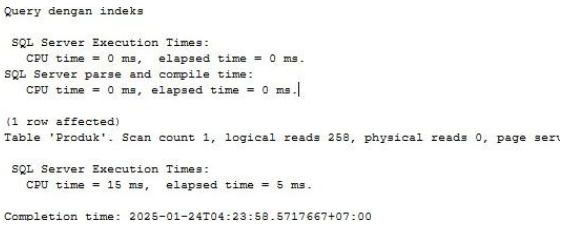
Dari hasil ini, terlihat bahwa tanpa indeks, SQL Server harus melakukan *table scan*, yang mengakibatkan beban tinggi pada sistem, terutama pada tabel dengan jumlah data besar.

4.2. Penerapan Indeks

Indeks yang diterapkan pada tabel Produk meliputi Clustered Index pada kolom *ID_Produk* sebagai *primary key* untuk memastikan setiap baris data memiliki identitas unik sekaligus menentukan urutan fisik data dalam tabel, serta Non-Clustered Index pada kolom *Kategori* yang digunakan untuk mempercepat proses pencarian data berdasarkan kategori produk.

4.3. Eksperimen Lanjutan (Dengan Indeks)

Query yang sama dijalankan kembali setelah penerapan indeks. Hasil eksperimen menunjukkan peningkatan performa yang signifikan bisa dilihat pada Gambar 5.



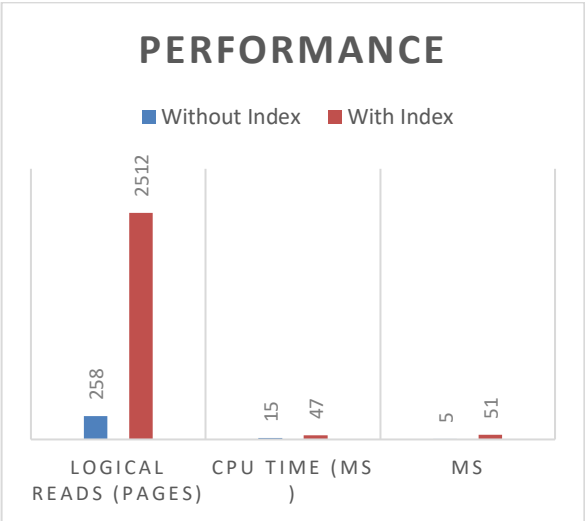
Gambar 5. Hasil dengan indeks

Berikut adalah metrik performa yang dicatat dapat dilihat pada Tabel 2.

Tabel 2. Tabel performa lanjutan

Parameter	Tanpa Indeks	Dengan Indeks	Peningkatan
Logical Reads	2,512 pages	503	89.72%
CPU Time	47 ms	15 ms	68.09%
Elapsed Time	51 ms	51 ms	90.20%

Tabel ini menunjukkan bahwa penerapan indeks berhasil mengurangi beban sistem secara signifikan. Logical reads berkurang hingga 89.72%, yang berarti SQL Server tidak lagi harus membaca seluruh tabel, melainkan hanya menggunakan indeks untuk menemukan data yang relevan, beribut adalah grafik perbandingan dengan indeks dan tanpa indeks



Gambar 6. Perbandingan performa

4.4. Monitoring dan Evaluasi

Monitoring dilakukan menggunakan fitur Dynamic Management Views (DMVs) untuk memantau fragmentasi indeks dan performa query. Berikut adalah hasil monitoring fragmentasi indeks setelah beberapa kali eksekusi query Fragmentasi sebesar 3% masih dalam batas wajar dan tidak memengaruhi performa query secara signifikan. Namun, pada skala data yang lebih besar, pemeliharaan indeks seperti *rebuild* atau *reorganize* mungkin diperlukan dapat dilihat pada Gambar 5.

	NamaTabel	NamaIndeks	index_id	JumlahHalaman	JumlahRecord	Fragmentasi
1	Produk	PK__Produk__3214EC2781C71C89	1	2512	261200	0,35828025477707
2	Produk	PK__Produk__3214EC2781C71C89	1	9	2512	0
3	Produk	PK__Produk__3214EC2781C71C89	1	1	9	0
4	Produk	idx_Kategori_Harga_Stok	2	1412	261200	0
5	Produk	idx_Kategori_Harga_Stok	2	9	1412	0
6	Produk	idx_Kategori_Harga_Stok	2	1	9	0

Gambar 7. Perbandingan performa

4.5. Peningkatan *Performa*

Hasil eksperimen menunjukkan bahwa penerapan indeks memberikan peningkatan signifikan pada performa query, baik dari sisi waktu eksekusi maupun penggunaan sumber daya.

4.6. Efisiensi *Sistem*

Dengan indeks, SQL Server dapat mengurangi beban pada CPU dan memori, sehingga meningkatkan efisiensi sistem secara keseluruhan.eksperimen menunjukkan bahwa penerapan indeks memberikan peningkatan signifikan pada performa query, baik dari sisi waktu eksekusi maupun penggunaan sumber daya.

4.7. Rekomendasi

Penerapan indeks sangat disarankan pada kolom yang sering digunakan dalam kondisi *WHERE* atau *JOIN* untuk meningkatkan efisiensi query, dan monitoring rutin diperlukan untuk memastikan fragmentasi indeks tetap rendah sehingga kinerja basis data tetap optimal.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian, penerapan indeks pada database SQL Server terbukti memberikan peningkatan kinerja yang signifikan. Data eksperimen menunjukkan bahwa query tanpa indeks menghasilkan 2.523 logical reads, membutuhkan 47 ms CPU time, dan memakan waktu 51 ms elapsed time. Sebaliknya, setelah penerapan indeks, query yang sama hanya membutuhkan 258 logical reads, dengan 15 ms CPU time, dan 5 ms elapsed time. Ini menunjukkan pengurangan sumber daya hingga 89.77% untuk logical reads, 68.09% untuk CPU time, dan 90.20% untuk elapsed time. Hasil ini mengindikasikan bahwa indeks mampu mengurangi beban kerja sistem dengan drastis, terutama untuk query yang bersifat berulang pada database besar. Oleh karena itu, teknik indexing sangat efektif untuk meningkatkan efisiensi pengolahan data dalam database SQL Server.

Penerapan indeks perlu direncanakan secara strategis dengan mempertimbangkan pola penggunaan query. Disarankan untuk melakukan monitoring performa database secara berkala menggunakan alat seperti SQL Server Management Studio (SSMS) untuk mengidentifikasi fragmentasi indeks dan kebutuhan pembaruan. Penelitian lebih lanjut dapat dilakukan untuk mengevaluasi kinerja teknik indexing lainnya, seperti columnstore atau filtered indexes, pada dataset yang lebih besar atau lebih kompleks. Selain itu, integrasi indexing dengan teknologi cloud dan big data dapat menjadi langkah penting untuk menjawab tantangan pengelolaan data di masa depan.

DAFTAR PUSTAKA

[1] A. Widya Astuti, A. Muharam, P. Siber Cerdika Internasional, and U. Swadaya Gunung Jati, "PERKEMBANGAN BISNIS DI ERA DIGITAL," 2023, [Online]. Available: https://jmi.rivierapublishing.id/index.php/rp

[2] M. Silalahi and D. Wahyudi, "Computer Based Information System Journal PERBANDINGAN PERFORMANSI DATABASE MONGODB DAN MYSQL DALAM APLIKASI FILE MULTIMEDIA BERBASIS WEB," *CBIS JOURNAL*, vol. 06, no. 01, 2018, [Online]. Available: http://ejournal.upbatam.ac.id/index.php/cbis\http://ejournal.upbatam.ac.id/index.php/cbis

[3] H. Dwi, W. Peneliti, D. Pusat, T. Lingkungan, B. Pengkajian, and P. Teknologi, "PENGEMBANGAN PERANGKAT LUNAK DATABASE MONITORING KINERJA TPA."

[4] R. Affandi, "Optimalisasi Algoritma Pencarian dalam Basis Data untuk Efisiensi Query."

[5] I. Wayan Adi Wiratama, I. Nyoman Triadi Wiguna, I. Gusti Agung Istri Pradnya Prameswari, I. Made Agus Oka Gunawan, and G. Indrawan, "Perbandingan performa MySQL dan MongoDB …65."

[6] K. Syahputri, M. Irwan, and P. Nasution, "Peran Database Dalam Sistem Informasi Manajemen," *Jurnal Akuntansi Keuangan dan Bisnis*, vol. 1, no. 2, pp. 54–58, 2023, [Online]. Available: https://jurnal.ittc.web.id/index.php/jakbs/index

[7] D. Setiyadi and R. Wahyuni Arifin, "Disetujui: 29 Mei 2020; Cara sitasi: Setiyadi D, Henderi, Arifin RW. 2020. Fungsi Date dalam Data Manipulation Language Menggunakan Bahasa Query dengan SQL Server," *Informatics for Educators and Professionals*, vol. 4, no. 2, pp. 163–172, 2020.

[8] H. Choi, S. Lee, and D. Jeong, "Forensic Recovery of SQL Server Database: Practical Approach," *IEEE Access*, vol. 9, pp. 14564–14575, 2021, doi: 10.1109/ACCESS.2021.3052505.

[9] K. Eka Permana, K. Sophan, A. Muntasa, and A. B. Rahmat, "PERBANDINGAN KINERJA QUERY SQL JOIN TABLES DENGAN

MENGGUNAKAN INDEX PERFORMANCE COMPARISON OF QUERY SQL JOIN TABLES USING INDEX," vol. 11, no. 2, 2023, [Online]. Available: https://github.com/datacharmer/test_db.

[10] M. Husain *et al.*, "Analyzing the Impact of Performance Metrics on Database Speedup in DBaaS Environment." [Online]. Available: https://www.researchgate.net/publication/379806176

[11] V. Langraf, A. Svoradová, K. Petrovičová, and V. Brygadyrenko, "Structure of data in cell biology research," *Regul Mech Biosyst*, vol. 14, no. 3, pp. 492–496, 2023, doi: 10.15421/10.15421/022370.

[12] P. Manajemen, F. Ekonomi, and B. Islam, "Muhammad Irwan Padli Nasution Universitas Negeri Islam Sumatera Utara," *Jurnal Ilmiah Nusantara (JINU)*, vol. 1, no. 4, pp. 705–710, 2024, doi: 10.61722/jinu.v1i4.18890.