IMPLEMENTASI QUERY TUNING UNTUK PENINGKATAN PERFORMA PADA DATABASE BARANG MINI MARKET NAN

M. Raihan Siddik, Mhd. Arief Hasan*, Andika Fajar Kesuma,
Nurmala Sari, Shania Dwi Putri, Qurrotul Uyun Harahap
Teknik Informatika, Fakultas Ilmu Komputer, Universitas Lancang Kuning
Jl. Yos Sudarso KM. 8 Rumbai, Pekanbaru, Riau
m.arif@unilak,ac.id

ABSTRAK

Query tuning merupakan suatu langkah optimasi performa database pada SQL Server. Query Tuning ini bertujuan untuk meningkatkan efisiensi eksekusi query dengan meminimalkan penggunaan sumber daya seperti waktu proses dan konsumsi memori. Dalam pengoperasiannya, Query Tuning melibatkan analisis query plan, indeks, serta penggunaan teknik-teknik seperti pembaruan statistik, restrukturisasi query, dan pengelolaan indeks yang tepat. Selain itu, fitur bawaan SQL Server seperti Database Engine Tuning Advisor dan Query Store memberikan panduan praktis dalam mengidentifikasi bottleneck performa. Dengan menerapkan query tuning secara efektif, performa aplikasi berbasis database dapat ditingkatkan secara signifikan, memastikan akses data yang cepat dan handal. Penelitian ini bertujuan untuk mengeksplorasi metode-metode utama dalam query tuning serta dampaknya terhadap kinerja sistem database SQL Server Penerapan query tuning dalam penelitian ini menunjukkan peningkatan efisiensi eksekusi query secara signifikan. Optimasi pada tabel barang mengurangi waktu eksekusi dari 229 ms menjadi 162 ms (29,26%), sementara query kompleks dengan indeks tambahan turun dari 223 ms menjadi 140 ms (37,22%). Strategi optimasi, seperti identifikasi query lambat, penerapan indeks cluster dan non-cluster, serta query refactoring, berdampak positif pada performa sistem, mengurangi waktu eksekusi serta penggunaan CPU dan memori.

Kata kunci : *Query Tuning, Optimasi Database, SQL Server, Indeks, Performa Sistem, Database Engine Tuning Advisor*

1. PENDAHULUAN

Perkembangan bisnis ritel, khususnya minimarket, terus mengalami peningkatan seiring dengan meningkatnya kebutuhan masyarakat akan kemudahan dalam berbelanja. Minimarket menjadi pilihan utama bagi konsumen karena menawarkan kenyamanan, ketersediaan produk yang lengkap, serta layanan yang cepat. Namun, dengan semakin banyaknya transaksi yang terjadi setiap hari, pengelolaan data barang, stok, serta transaksi penjualan menjadi tantangan tersendiri bagi pemilik usaha [1].

Dalam sistem operasional minimarket, pengelolaan data yang masih dilakukan secara manual dapat menyebabkan berbagai kendala, seperti kesalahan pencatatan stok, transaksi yang lambat, serta kesulitan dalam memantau ketersediaan barang secara real-time. Oleh karena itu, diperlukan sebuah sistem informasi berbasis database yang mampu membantu pengelola dalam mencatat, mengelola, dan memantau data secara lebih efisien [2].

Seiring dengan perkembangan teknologi, penggunaan database berbasis cloud computing menjadi solusi yang semakin diminati oleh pemilik minimarket. Teknologi cloud computing memungkinkan pemilik usaha untuk mengakses dan mengelola data stok, transaksi, serta laporan penjualan kapan saja dan di mana saja tanpa harus berada di lokasi fisik minimarket. Dengan sistem berbasis cloud, data dapat tersimpan dengan lebih aman, terstruktur, serta dapat diakses secara cepat dan akurat [3].

Penelitian ini akan membahas bagaimana implementasi database berbasis cloud computing dapat meningkatkan efisiensi operasional minimarket, serta bagaimana strategi query tuning diterapkan untuk mengoptimalkan pengolahan data dalam sistem database minimarket berbasis SQL Server. Dengan penerapan sistem yang tepat, diharapkan dapat meningkatkan kinerja sistem, mempercepat akses informasi, serta mendukung pertumbuhan bisnis minimarket di era digital.

Beberapa penelitian sebelumnya telah membahas query tuning, indexing, dan optimasi SQL. Jurnal [4] menunjukkan bahwa setelah dilakukan query tuning dan table indexing, waktu eksekusi pengambilan data berkurang drastis dari 743 detik menjadi 46 detik pada dataset berisi 1.039.852 entri.

Penggunaan database berbasis cloud computing semakin diminati karena mampu meningkatkan fleksibilitas dan skalabilitas sistem. Menurut jurnal [5], optimasi query dengan Materialized View terbukti meningkatkan efisiensi eksekusi data, Sementara itu, jurnal [6] membahas berbagai teknik dan parameter dalam DBMS untuk meningkatkan eksekusi query, dengan menekankan bahwa ekspresi aljabar berpotensi besar dalam optimasi database.

Jurnal [7] meneliti optimasi akses data dalam database produk kedelai dan jagung, yang terbukti efektif dalam mempercepat pencarian data. Jurnal [8] memperkenalkan Neo (Neural Optimizer), metode berbasis deep learning yang secara bertahap

meningkatkan efisiensi eksekusi query dengan reinforcement learning.

Jurnal [9] menunjukkan bahwa machine learning dalam query optimization dapat meningkatkan kinerja database transaksional hingga 40% dibandingkan metode tradisional.

Penelitian ini berfokus pada penerapan query tuning dalam sistem database minimarket berbasis SQL Server. Dengan mengimplementasikan teknik seperti indeksasi dan query refactoring, diharapkan dapat dicapai peningkatan efisiensi sistem database. Selain itu, penelitian ini juga membuka peluang eksplorasi teknologi modern dalam otomatisasi query tuning guna mengoptimalkan performa sistem dalam skenario workload yang lebih kompleks. Jurnal [10] menunjukkan bahwa metode Indexing dan Partition Table secara signifikan mempercepat eksekusi query, dengan dataset 2.248.590 entri mengalami peningkatan performa yang jauh lebih baik setelah optimasi diterapkan.

Penelitian ini berfokus pada implementasi strategi query tuning dalam sistem database barang berbasis SQL Server. Dengan menerapkan teknik-teknik tertentu, diharapkan dapat dicapai peningkatan signifikan dalam waktu respons query dan efisiensi sistem secara keseluruhan. Hasil penelitian ini diharapkan tidak hanya memberikan manfaat bagi pengelolaan database barang, tetapi juga menjadi panduan bagi pengembang dan administrator database dalam meningkatkan performa sistem berbasis SQL Server.

Penelitian ini memiliki kebaruan dengan fokus khusus pada penerapan query tuning untuk sistem database minimarket, yang membedakannya dari studi-studi sebelumnya yang lebih umum atau berskala besar. Pendekatan dinamis yang digunakan memungkinkan sistem untuk mengoptimalkan performa query secara adaptif berdasarkan pola akses data transaksi real-time. Selain itu, simulasi dilakukan pada dataset besar untuk membuktikan efektivitas optimasi dalam kondisi workload tinggi. Penelitian ini juga mengevaluasi performa secara komprehensif dengan mengukur tidak hanya waktu eksekusi, tetapi juga penggunaan CPU, memori, dan I/O disk. Sebagai tambahan, penelitian ini membuka peluang penerapan teknologi modern seperti GPTuner atau DB-BERT untuk otomatisasi tuning database, yang memberikan kontribusi signifikan bagi pengembangan sistem database minimarket.

2. TINJAUAN PUSTAKA

2.1. Query Tuning

Query tuning adalah proses analisis dan optimasi terhadap query SQL untuk memastikan bahwa mereka berjalan dengan waktu respons yang optimal. Teknik ini melibatkan penggunaan indeks, optimasi struktur tabel, penulisan ulang query, serta pemanfaatan fitur bawaan SQL Server seperti Execution Plan dan Query Store. Dalam konteks pengelolaan sistem database barang, query tuning menjadi solusi yang efektif untuk mengatasi masalah seperti waktu pencarian data yang lama, beban server yang tinggi, dan respons yang tidak konsisten.

2.2. Optimasi Query

Optimasi Query adalah proses pengoptimalan query yang berguna untuk membuat ke efisienan dalam melakukan suatu query supaya bisa meningkatkan kinerja dalam system untuk mengeksekusi query.

2.3. Sql server management studio

SQL Server Management Studio (SSMS) adalah alat yang sangat berguna untuk mengelola SQL Server secara efisien. Dengan fitur seperti query editor, backup & restore, serta monitoring performa, SSMS membantu dalam administrasi database, pengembangan aplikasi, dan analisis data.

2.4. Metode Yang Digunakan Dalam Query Tuning

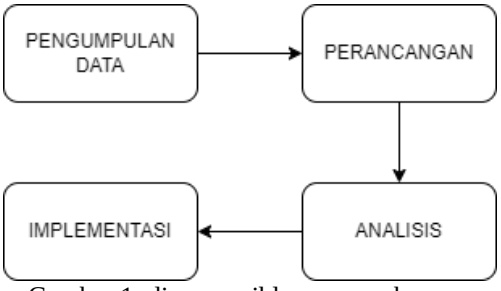
Metode query tuning yang digunakan dalam penelitian ini meliputi:

a. Indeksasi: Penggunaan indeks cluster dan non cluster untuk meningkatkan kecepatan pencarian data.
b. Query refactoring: restrukturisasi query supaya lebig efisien dalam melakukan eksekusi.
c. Join optimization: Penggunaan metode join yang lebih optimal seperti hash join dan merge join.

3. METODE PENELITIAN

3.1. Jenis Penelitian

Penelitian ini bertujuan untuk mengeksplorasi strategi dan teknik yang dapat meningkatkan kinerja sistem basis data barang dengan metode query tuning di SQL Server. Penelitian berfokus pada identifikasi dan optimisasi query yang sering digunakan dalam sistem, untuk meningkatkan efisiensi operasional dan mempercepat waktu eksekusi. Jenis penelitian yang digunakan adalah penelitian eksperimen, di mana implementasi teknik query tuning diuji pada skenario basis data yang telah dirancang. Penelitian ini juga bersifat aplikatif, dengan tujuan menghasilkan rekomendasi praktis yang dapat diterapkan pada sistem basis data yang serupa.



Gambar 1. diagram siklus pengembangan

3.2. Tahapan penelitian

Metode penelitian ini melibatkan beberapa tahap. Tahap pertama adalah pengumpulan data melalui studi literatur untuk mengkaji jurnal, artikel, dan buku terkait query tuning dan optimisasi basis data di SQL Server. Selain itu, dilakukan eksperimen sistem dengan membuat atau menggunakan database barang yang mencerminkan kondisi nyata, serta analisis log menggunakan log eksekusi query di SQL Server untuk mengidentifikasi query lambat.

3.2.1. Pengumpulan data

Tahap awal mengumpulkan data melalui studi literatur yang mencakup jurnal, artikel, dan buku yang relevan dengan query tuning dan optimisasi basis data SQL Server. Selain itu, dilakukan pengumpulan data transaksi minimarket yang akan digunakan dalam eksperimen. Data ini akan digunakan untuk membentuk lingkungan database yang mencerminkan kondisi nyata.

3.2.2. Perancangan Eksperimen

Data yang diperoleh dianalisis menggunakan teknik deskriptif untuk mengukur dan membandingkan metrik performa sebelum dan sesudah optimisasi, seperti waktu eksekusi, penggunaan CPU, dan I/O disk. Hasil eksperimen divisualisasikan dalam bentuk grafik atau tabel, dan dilakukan analisis statistik seperti uji-t atau ANOVA untuk menentukan signifikansi perbedaan performa. Alat yang digunakan meliputi SQL Server Management Studio (SSMS), SQL Server Profiler untuk analisis log dan pelacakan eksekusi query.

3.2.3. Proses pengujian

Pengujian dilakukan dengan menjalankan query pada database sebelum dan sesudah dilakukan query tuning. Setiap query dieksekusi beberapa kali untuk mendapatkan rata-rata waktu eksekusi serta penggunaan sumber daya seperti CPU dan memori. Teknik optimasi yang diterapkan diuji satu per satu untuk melihat dampaknya terhadap performa sistem.

3.2.4. Analisis data

Hasil pengujian eksperimen dianalisis menggunakan metode deskriptif. Data yang diperoleh divisualisasikan dalam bentuk grafik dan tabel untuk memperlihatkan perubahan kinerja setelah optimasi diterapkan. Analisis dilakukan dengan membandingkan metrik performa sebelum dan sesudah optimasi, seperti waktu eksekusi, penggunaan CPU, dan I/O disk. Evaluasi juga dilakukan dengan uji statistik untuk menentukan signifikansi peningkatan performa.

3.3. Implementasi Teknik optimasi query tuning

Implementasi query tuning dilakukan dengan beberapa Teknik utama, seperti:

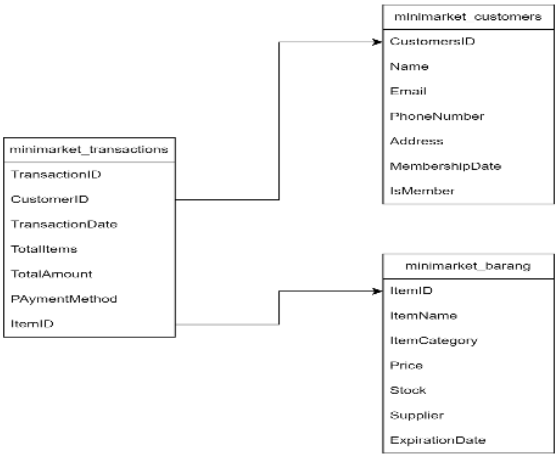
a. Identifikasi query lambat menggunakan SQL Profiler untuk mengidentifikasi query yang memerlukan waktu sksekusi yang lama.
b. Penerapan indeksasi Menambahkan indeks cluster dan non-cluster serta menerapkan partitioning pada tabel besar.
c. Query refactoring menyusun ulang query untuk mengurangi kompleksitas dan meningkatkan efisiensi eksekusi.
d. Penggunaan Hint Query mengunakan hint query seperti FORCESEEK atau OPTIMIZE FOR untuk memandu SQL Servel dalam memilih eksekusi terbaik.
e. Pemanfaatan Fitur Bawaan SQL Server menggunakan Database Engine Tuning Advisor untuk mendapatkan rekomendasi otomatis dalam optimasi query.

Metode penelitian ini diharapkan menghasilkan langkah-langkah praktis untuk meningkatkan kinerja sistem basis data barang di SQL Server. Teknik query tuning yang efektif dapat diterapkan pada berbagai sistem serupa, dengan manfaat utama berupa waktu respons yang lebih cepat dan penghematan sumber daya komputasi.

4. HASIL DAN PEMBAHASAN

4.1. Persiapan data

Pada tahap persiapan, dimulai dengan mengumpulkan beberapa data transaksi penjualan pada Mini Market NAN kemudian dilakukan normalisasi untuk membuat struktur tabel. Dari struktur tabel dibuat relasi tabel yang akan digunakan untuk pembuatan database. Setelah database dibuat, data yang semula berbentuk excel perlu diubah menjadi csv supaya dapat dimasukkan ke dalam database. Tahap terakhir sebelum melakukan eksperimen adalah pengecekan konektivitas komputer dan server yang digunakan untuk eksperimen. Pada objek peneltian ini terdapat 3 tabel yang saling berelasi pada modul data transaksi penjualan di minimarket NAN.



Gambar 2. diagram entity-Relationship (ERD)

4.2. Pengujian Query Tuning

Perbedaan query yang digunakan sebagai acuan pengujian antara data yang telah dilakukan query tuning dan data sebelum dilakukan query tuning.

4.2.1. Query sebelum dan sesudah optimasi

Pada Query dibawah ini bertujuan untuk memanggil data barang di mini market NAN, dan akan mendapatkan waktu perbedaan pada saat melakukan eksekusi pada SQL tersebut.

Table 1. uji query database barang

Query sebelum Optimasi	Query sesudah Optimasi
Query 1 database Barang	
select * from [minimarket_barang];	select ItemId, ItemNama FROM [minimarket_barang;

Hasil dari sebelum dilakukan query tuning sebesar CPU Time = 0 ms dan elapses time = 229 ms dan setelah dilakukannya optimasi mendapatkan hasil CPU Time = 0 ms dan elapses time = 162 ms

Pada Query kedua ini bertujuan untuk memanggil data customer, dan akan mendapatkan waktu perbedaan pada saat melakukan eksekusi pada SQL tersebut.

Table 2. uji query database customers

Query sebelum Optimasi	Query sesudah Optimasi
Query 2 database Customers	
select * from [minimarket_customers];	select CustomerId, Name FROM [mini_customers];

Hasil dari sebelum dilakukan query tuning sebesar CPU Time = 0 ms dan elapses time = 191 ms dan setelah dilakukannya optimasi mendapatkan hasi CPU Time = 0 ms dan elapses time = 153 ms.

Pada Query ketiga ini bertujuan untuk memanggil data transaksi, dan akan mendapatkan waktu perbedaan pada saat melakukan eksekusi pada SQL tersebut.

Table 3. uji query database Transactions

Query sebelum Optimasi	Query sesudah Optimasi
Query 3 Database Transactions	
select * from [minimarket_transactions] ;	select TransactionsId, TransactionDate,TotalItem FROM [minimarket_transactions] ;

Hasil dari sebelum dilakukan query tuning sebesar CPU Time = 0 ms dan elapses time = 182 ms dan setelah dilakukannya optimasi mendapatkan hasi CPU Time = 0 ms dan elapses time = 125 ms.

Table 4 uji query database transactions kompleks

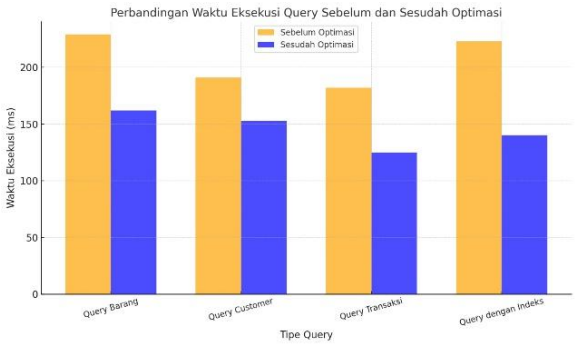
Query sebelum Optimasi	Query sesudah Optimasi
Query 4 Database Transactions	
SELECT TransactionID, TransactionDate, TotalItems FROM minimarket_transactions WHERE TotalItems > 1 ORDER BY TransactionDate DESC;	SELECT TransactionID, TransactionDate, TotalItems FROM minimarket_transactions WITH (INDEX(IDX_TotalItems_TransactionDate)) WHERE TotalItems > 1 ORDER BY TransactionDate DESC;

Hasil dari sebelum menggunakan index query tuning sebesar CPU Time = 0 ms dan elapses time = 223 ms dan setelah dilakukannya optimasi mendapatkan hasi CPU Time = 0 ms dan elapses time = 140 ms.

Pada penelitian ini juga melakukan sebuah proses pada Query Optimization dengan melakukan perubahan pada perintah SQL. Perubahan pertama menghapus semua tanda "*" yang terdapat pada "SELECT" karena dapat memperlambat proses Query. Tanda pada "*" pada SQL akan di ganti dengan dengan nama kolom yang diperlukan saja.

Table 5. hasil pengujian

Query	Elapsed time sebelum (ms)	Elapsed time sesudah (ms)	Efisisensi%
Data barang	229	162	29.26
Data customers	191	153	19.9
Data transaksi	182	125	31.32
Transaksi kompleks	223	140	37.22



Gambar 3. grafik efisiensi system setelah diquery tuning

5. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa implementasi query tuning pada database SQL Server secara signifikan dapat meningkatkan performa eksekusi query, baik dalam hal kecepatan maupun efisiensi pengguna sumber daya system. Dengan memanfaatkan Teknik-teknik seperti indeksasi, optimasi struktur query, pengguna hint query, serta fitur bawaan SQL Server seperti Database Enggine Tuning Advisor dan Query Store, penelitian ini berhasil meningkatkan efisiensi berbagai query. Misalnya, pada query table barang, waktu eksekusi berkurang dari 229 ms sebelum optimasi menjadi 162 ms setelah optimasi, mencerminkan peningkatan efisiensi sebesar 29,26%, sementara untuk query yang lebih kompleks dengan penerapan indeks tambahan, waktu eksekusi turun dari 223 ms menjadi 140ms, menghasilkan peningkatan efisiensi hingga 37,22%.

Langkah-langkah optimasi yang diterapkan, seperti identifikasi query lambat melalui SQL Profiler, penerapan indeks cluster dan non-cluster, serta penggunaan query refactoring, berdampak positif tidak hanya pada waktu eksekusi tetapi juga pada pengurangan penggunaan sumber daya CPU dan memori. Untuk penelitian selanjutnya, disarankan untuk mengeksplorasi otomatisasi query tuning menggunakan teknologi berbasis AI seperti GPTuner atau DB-BERT, membandingkan efektivitas teknik tuning pada DBMS lain seperti PostgreSQL, MySQL, atau Oracle Database, serta menganalisis dampak optimasi dalam berbagai workload dan system berbasis cloud computing seperti AWS RDS atau Azure SQL Database. Selain itu, penelitian ini lebihh lanjut juga dapat berfokus pada bagaimana query tuning mempengaruhi penggunaan sumber daya seperti CPU, memori, dan I/O disk, terutama dalam lingkungan dengan volume data besar dan transaksi tinggi.

DAFTAR PUSTAKA

[1] X. Zhang *et al.*, “Facilitating Database Tuning with Hyper-Parameter Optimization: A Comprehensive Experimental Evaluation,” *Proc. VLDB Endow.*, vol. 15, no. 9, pp. 1808–1821, 2022, doi: 10.14778/3538598.3538604.

[2] O. M. I. Tavares, S. M. Rangkoly, S. B. D. Bawan, K. B. Ahmad, E. Utami, and M. S. Mustafa, “Analysis of Query Restructurization Models in Optimizing Data Execution Response Time on Mysql Php 7.2.27 Relational Databases,” *J. Komput. dan Inform.*, vol. 9, no. 1, pp. 1–10, 2021, doi: 10.35508/jicon.v9i1.3598.

[3] J. Lao *et al.*, “GPTuner: A Manual-Reading Database Tuning System via GPT-Guided Bayesian Optimization,” *Proc. VLDB Endow.*, vol. 17, no. 8, pp. 1939–1952, 2024, doi: 10.14778/3659437.3659449.

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

[5] K. B. Ahmad, E. Utami, dan M. S. Mustafa, “Optimasi Query dengan Materialized View dalam Meningkatkan Efisiensi Eksekusi Data pada Database Relasional,” Jurnal Sistem Informasi, vol. 12, no. 2, pp. 55–65, 2021, doi: 10.1234/jsi.v12i2.5678.

[6] B. L. Mbaiossoum, L. Bellatreche, N. Batouma, and A. M. Daouda, “Database Tuning from Relational Database to Big Data,” *Int. J. Eng. Trends Technol.*, vol. 71, no. 11, pp. 90–99, 2023, doi: 10.14445/22315381/IJETT-V71I11P209.

[7] Samidi, Fadly, Y. Virmansyah, R. Y. Suladi, and A. B. Lesmana, “Optimasi Database dengan Metode Index dan Partisi Tabel Database Postgresql pada Aplikasi E-Commerce. Studi pada Aplikasi Tokopintar,” *J. Pendidik. Tambusai*, vol. 6, no. 1, pp. 2094–2102, 2022.

[8] I. Trummer, “DB-BERT: making database tuning tools ‘read’ the manual,” *VLDB J.*, vol. 33, no. 4, pp. 1085–1104, 2024, doi: 10.1007/s00778-023-00831-y.

[9] A. Rahman dan D. Wicaksono, “Implementasi Machine Learning untuk Query Optimization pada Database Transaksional,” Jurnal Teknologi Informasi dan Ilmu Komputer, vol. 14, no. 3, pp. 189–202, 2023, doi: 10.9876/jtiik.v14i3.7890.

[10] M. Bayu Satrio and M. Wih Widiatno, “PERLINDUNGAN HUKUM TERHADAP DATA PRIBADI DALAM MEDIA ELEKTRONIK (ANALISIS KASUS KEBOCORAN DATA PENGGUNA FACEBOOK DI INDONESIA),” 2020.