ANALISIS EFEKTIVITAS TEKNIK *INDEXING* TERHADAP WAKTU EKSEKUSI *QUERY* SQL PADA BASIS DATA TRANSAKSI *E-COMMERCE*

M.Aldy Zulfa Saputra[1)], Muchammad Soufwan Fauzi[2)], Rangga Dzikri Fardiansyah[3)], Zaki Kafila Pamungkas[4)], Dicky Anggriawan Nugroho[5)], Imam Prayogo Pujiono[6)]

Prodi Informatika, Fakultas Ekonomi dan Bisnis Islam, UIN K.H Abdurrahman Wahid Pekalongan

Correspondence author: MAZ Saputra, m.aldy.zulfa.saputra24055@mhs.uingusdur.ac.id, Pekalongan, Indonesia

Abstract

This study examines the effect of indexing techniques on SQL query execution speed in a MySQL-based e-commerce transaction database. Performance issues often arise as data volume increases because the system must scan the entire table, resulting in slow response times. To analyze this problem, this study employed a pure experimental method by testing three index conditions: no index, a single index, and a composite index. Testing was conducted using a synthetic dataset of 20,000 rows depicting e-commerce transaction activity. Each condition was evaluated using a series of SELECT queries. Measurements were made through execution time and execution plans displayed by the EXPLAIN command. The results showed that using indexes provided a significant performance improvement. A single index successfully reduced execution time from 0.2380 seconds to 0.0740 seconds, while a composite index resulted in an execution time of 0.0900 seconds. Both types of indexes reduced the number of rows scanned across the entire table to only tens of rows. A single index is more suitable for simple queries with high selectivity, while a composite index is more effective for queries with more than one condition. However, implementing an index requires additional storage space and causes the data write process to run slightly slower. Overall, this study confirms that an indexing strategy tailored to query patterns and data characteristics can significantly improve the performance of e-commerce transaction databases.

Keyword *: effectiveness analysis, indexing, SQL queries, databases, e-commerce*

Abstrak

Penelitian ini mengkaji pengaruh teknik *indexing* terhadap kecepatan eksekusi *query* SQL pada basis data transaksi *e-commerce* berbasis MySQL. Gangguan kinerja sering muncul ketika jumlah data meningkat karena sistem harus melakukan pemindaian seluruh tabel yang menyebabkan waktu respon menjadi lambat. Untuk menganalisis permasalahan tersebut, penelitian ini menerapkan metode eksperimen murni dengan menguji tiga kondisi indeks yaitu tanpa indeks, indeks tunggal, dan indeks komposit. Pengujian dilakukan menggunakan *dataset* sintetis berjumlah 20.000 baris yang menggambarkan aktivitas transaksi *e-commerce*. Setiap kondisi dievaluasi menggunakan serangkaian *query* SELECT.

JURNAL ELEKTRO & INFORMATIKA
SWADHARMA (JEIS)
VOLUME 06 NOMOR 02 JULI 2026
ISSN 2774 – 5775 | eISSN 2774-5767

Pengukuran dilakukan melalui waktu eksekusi dan rencana eksekusi yang ditampilkan oleh perintah EXPLAIN. Hasil penelitian menunjukkan bahwa penggunaan indeks memberikan peningkatan performa yang sangat signifikan. Indeks tunggal berhasil menurunkan waktu eksekusi dari 0,2380 detik menjadi 0,0740 detik, sedangkan indeks komposit menghasilkan waktu eksekusi 0,0900 detik. Kedua jenis indeks ini mengurangi jumlah baris yang dipindai dari seluruh tabel menjadi hanya puluhan baris. Indeks tunggal lebih sesuai untuk *query* sederhana dengan *selectivity* yang tinggi, sedangkan indeks komposit lebih efektif untuk *query* yang memiliki lebih dari satu kondisi. Meskipun demikian, penerapan indeks memerlukan ruang penyimpanan tambahan dan menyebabkan proses penulisan data berjalan sedikit lebih lambat. Secara keseluruhan, penelitian ini menegaskan bahwa strategi *indexing* yang disesuaikan dengan pola *query* dan karakteristik data mampu meningkatkan performa basis data transaksi *e-commerce* secara signifikan.

Kata Kunci : analisa efektivitas, *indexing*, *query* sql, basis data, *e-commerce*

A. PENDAHULUAN

Perkembangan teknologi informasi telah menyebabkan hampir seluruh aktivitas manusia terekam dalam bentuk data, mulai dari interaksi pada media sosial, layanan perbankan, hingga berbagai transaksi digital. Kebutuhan akan pengelolaan data dalam jumlah besar tersebut menuntut tersedianya sistem yang mampu menyimpan, mengelola, dan mengolah informasi dalam volume besar secara cepat, tepat, dan andal (Ibna, 2024). Tanpa pengelolaan basis data yang efisien, proses penyajian informasi, pengambilan keputusan, serta layanan berbasis aplikasi berpotensi menjadi lambat dan tidak responsif. Kondisi ini semakin krusial pada aplikasi yang bergantung pada pemrosesan data secara waktu nyata *(real-time)*. Salah satu di antaranya ialah aplikasi *e-commerce* yang tidak hanya berfungsi sebagai sarana fasilitasi transaksi jual beli secara daring, tetapi juga sebagai sistem informasi yang menganalisis data konsumen, riwayat transaksi, dan perilaku pengguna sebagai dasar perumusan kebijakan dan pengambilan keputusan bisnis (Rahayu et al., 2025). Dengan demikian, kebutuhan akan sistem basis data yang mampu menangani volume data besar serta mengelola data secara efisien menjadi fondasi penting dalam pengembangan dan pengelolaan aplikasi *e-commerce* modern.

Dalam praktiknya, salah satu teknologi yang banyak digunakan untuk mengelola data transaksi pada aplikasi *e-commerce* adalah *Relational Database Management System* (RDBMS). RDBMS menyimpan data dalam bentuk tabel-tabel berelasi dan dikelola menggunakan *Structured Query Language* (SQL) sebagai bahasa standar untuk mendefinisikan dan mengakses data (Adisa & Nasution, 2023). Contoh RDBMS yang umum digunakan antara lain *MySQL*, *PostgreSQL*, dan *Microsoft SQL Server* (Ahsana et al., 2023). Di antara berbagai pilihan tersebut, *MySQL* menjadi salah satu RDBMS yang populer karena bersifat sumber terbuka (*open source*), menggunakan SQL sebagai bahasa utama, serta banyak diimplementasikan pada sistem informasi dan aplikasi web, termasuk sistem penjualan dan *e-commerce* (Nurhayati & Nasution, 2023). Namun, seiring meningkatnya volume data dan kompleksitas kebutuhan informasi, kinerja eksekusi *query* pada MySQL dapat menurun apabila tidak disertai perancangan struktur basis data dan penerapan teknik optimasi yang memadai, seperti pemanfaatan

indexing untuk menjaga efisiensi waktu respon *query* (Thoib et al., 2024).

Dalam konteks tersebut, platform *e-commerce* menjadi salah satu contoh nyata aplikasi yang sangat bergantung pada kinerja sistem basis data. Peningkatan volume transaksi daring, personalisasi layanan, integrasi dengan sistem pembayaran digital, serta pemanfaatan analitik data berskala besar menghasilkan lonjakan data transaksi yang harus diproses secara cepat dan konsisten. Kondisi ini menuntut sistem basis data mampu mengeksekusi *query* dengan waktu tanggap yang rendah, terutama pada skenario transaksi yang berlangsung secara waktu nyata *(real-time)* (Anchlia, 2024). Keterlambatan beberapa detik saja dalam pemrosesan *query* dapat berdampak pada terhambatnya proses bisnis, penurunan kualitas layanan, hingga hilangnya peluang transaksi pada platform *e-commerce* (Poggi et al., 2014).

Menanggapi permasalahan tersebut, salah satu pendekatan yang banyak digunakan adalah penerapan teknik *indexing*. Secara konsep, indeks merupakan struktur data yang dibangun di atas satu atau beberapa kolom tabel untuk mempercepat proses pencarian dan penyaringan data, sehingga sistem tidak perlu melakukan memindai semua tabel (*full table scan)* pada setiap eksekusi *query* (Anchlia, 2024). Dengan adanya indeks, basis data dapat langsung mengarah ke lokasi data yang relevan sehingga jumlah blok data yang harus diakses menjadi lebih sedikit dan waktu respon *query* dapat ditekan secara signifikan. Berbagai penelitian terdahulu menunjukkan bahwa *indexing* merupakan salah satu teknik yang paling efektif dalam meningkatkan kinerja *query* pada sistem basis data relasional. (Anchlia, 2024) menegaskan bahwa pemanfaatan indeks yang tepat mampu menurunkan waktu pencarian data dan mengoptimalkan rencana eksekusi *query*, sedangkan (Panggabean & Azizah, 2025) menunjukkan bahwa penambahan indeks pada kolom-kolom transaksi tertentu dalam aplikasi *e-commerce* dapat mempercepat eksekusi *query* dan meningkatkan efisiensi pencarian data. Penelitian lain pada MySQL yang mengkaji optimasi *query* berbasis indeks juga melaporkan bahwa pemilihan kolom dan jenis indeks yang tepat berkontribusi langsung terhadap perbaikan waktu respon pada berbagai skenario pengujian (Maesaroh et al., 2022; Permana et al., 2023). Meskipun demikian, sebagian besar studi tersebut masih berfokus pada pengujian umum atau konteks basis data tertentu dan belum secara spesifik menganalisis perbandingan efektivitas desain indeks tanpa indeks, *single index*, dan *composite index* terhadap waktu eksekusi *query* SQL pada basis data transaksi *e-commerce* dengan volume data puluhan ribu baris menggunakan MySQL. Kesenjangan inilah yang menjadi dasar perlunya penelitian ini untuk menghadirkan kajian empiris yang lebih terarah pada skenario transaksi *e-commerce* yang mendekati kondisi operasional nyata.

Berdasarkan kesenjangan penelitian tersebut, fokus kajian dalam penelitian ini diarahkan pada analisis efektivitas penerapan teknik *indexing* terhadap kinerja eksekusi *query* pada basis data transaksi *e-commerce* berbasis MySQL. Pertanyaan penelitian yang diajukan adalah: (1) bagaimana pengaruh penerapan teknik *indexing* terhadap waktu eksekusi *query* SQL pada basis data transaksi *e-commerce*; (2) seberapa besar perbedaan performa *query* antara kondisi tanpa indeks, *single index*, dan *composite index*; dan (3) jenis indeks apa yang paling efektif digunakan untuk meningkatkan efisiensi waktu eksekusi *query* pada sistem *e-commerce* berskala besar. Sejalan dengan rumusan masalah tersebut, penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja waktu eksekusi *query* pada ketiga skenario desain indeks tersebut menggunakan basis data transaksi *e-commerce* berukuran puluhan ribu baris di MySQL, serta merumuskan rekomendasi strategi *indexing* yang dapat dimanfaatkan sebagai acuan bagi pengembang dan

pengelola basis data dalam mengoptimalkan performa sistem *e-commerce*.

B. METODE PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif eksperimental dengan jenis eksperimen murni (*true experiment*) untuk mengukur secara empiris pengaruh penerapan teknik *indexing* terhadap waktu eksekusi *query* SQL pada basis data transaksi *e-commerce*. Variabel independen dalam penelitian ini adalah skenario penerapan indeks, yang terdiri atas tiga kondisi, yaitu tanpa indeks, *single index*, dan *composite index*, sedangkan variabel dependen adalah waktu eksekusi *query*. Rancangan penelitian mengikuti model *pre-test–post-test control design*, di mana kinerja *query* diukur terlebih dahulu pada kondisi tanpa indeks, kemudian dibandingkan dengan kondisi setelah penerapan *single index* dan *composite index* sebagaimana digunakan pada penelitian eksperimen basis data oleh (Anchlia, 2024) dan (Panggabean & Azizah, 2025).

Penelitian dilaksanakan menggunakan satu perangkat laptop dengan sistem operasi Windows 11, prosesor Intel Core i7 Gen 12, RAM 16 GB, dan SSD 512 GB. Sistem manajemen basis data yang digunakan adalah MySQL Server 8.0.36 yang diakses melalui phpMyAdmin 5.2.1 dengan dukungan XAMPP sebagai server lokal. Populasi penelitian adalah data transaksi *e-commerce* yang memuat informasi pelanggan, produk, dan aktivitas pembelian. Dari populasi tersebut diambil sampel berupa *synthetic dataset* berukuran 20.000 baris yang diimpor ke dalam tabel *sales_transactions* dengan kolom utama *Customer_Id*, *Product_Category*, *Order_Date*, dan *Order_Status*. Pemilihan ukuran sampel ini dimaksudkan untuk merepresentasikan pola transaksi *e-commerce* pada volume data menengah yang relevan untuk pengujian kinerja basis data.

Teknik sampling yang digunakan adalah *purposive sampling* dengan memilih kolom dan pola *query* yang relevan dengan aktivitas pencarian data pada sistem *e-commerce*, seperti pencarian transaksi berdasarkan pelanggan, kategori produk, dan rentang waktu pemesanan. Penelitian diawali dengan pembuatan skema basis data dan pengisian tabel *sales_transactions* menggunakan *dataset* tersebut. Selanjutnya disusun serangkaian *query* SELECT yang mewakili pola akses data umum. Pada tahap pertama, seluruh *query* dieksekusi pada kondisi tanpa indeks untuk memperoleh kinerja dasar. Tahap berikutnya adalah penerapan *single index* pada kolom *Customer_Id*, diikuti pengujian ulang *query* yang sama. Tahap terakhir adalah penerapan *composite index* pada kombinasi kolom *Customer_Id* dan *Order_Date*, kemudian dilakukan pengujian ulang. Pada setiap skenario, perintah EXPLAIN digunakan untuk menganalisis rencana eksekusi *query* dan pola akses data.

Data yang diperoleh berupa waktu eksekusi *query* dan informasi rencana eksekusi dianalisis secara kuantitatif deskriptif. Analisis dilakukan dengan menghitung rata-rata, nilai minimum, maksimum, dan simpangan baku waktu eksekusi untuk setiap jenis *query* pada masing-masing skenario indeks, serta membandingkan kinerja antar skenario melalui persentase peningkatan atau penurunan waktu eksekusi. Hasil analisis kemudian disajikan dalam bentuk tabel, grafik, dan uraian naratif yang menjelaskan efektivitas masing-masing skenario indeks terhadap efisiensi waktu eksekusi *query* pada basis data transaksi *e-commerce*. Keabsahan hasil dijaga melalui replikasi pengujian pada setiap skenario dan pengendalian lingkungan eksekusi, dengan memastikan tidak ada proses lain yang membebani server selama eksperimen berlangsung sehingga perubahan kinerja terutama disebabkan oleh perbedaan konfigurasi indeks, sejalan dengan prinsip validitas internal pada studi optimasi basis data (Taipalus, 2024).

C. HASIL DAN PEMBAHASAN

Desain Indeks

Desain indeks dalam penelitian ini dilakukan melalui pendekatan sistematis dengan menentukan kolom yang akan diindeks berdasarkan analisis pola *query* pada *dataset* transaksi *e-commerce* yang berjumlah 20.000 baris. Penggunaan data dengan volume menengah ini dirancang untuk merepresentasikan kondisi operasional sistem *e-commerce* yang umum digunakan pada skala UMKM hingga bisnis daring berkembang. Studi terkait oleh (Panggabean & Azizah, 2025) juga menerapkan *dataset* transaksi berukuran serupa dalam penelitian performa *query* sehingga ukuran data ini dinilai representatif.

Secara teoretis, konsep *selectivity* menjadi dasar penting dalam pemilihan kolom yang akan diberikan indeks. *Selectivity* merujuk pada tingkat keunikan nilai dalam suatu kolom; kolom dengan *selectivity* tinggi (mendekati 1) berisi banyak nilai unik, sedangkan kolom dengan *selectivity* rendah (mendekati 0) didominasi oleh nilai yang berulang. Berdasarkan teori yang dikemukakan oleh (Sakshi & Sharma, 2025), kolom dengan *selectivity* tinggi lebih optimal untuk diindeks karena dapat mempersempit ruang pencarian secara signifikan dan mengurangi jumlah baris yang harus dipindai selama eksekusi *query*.

Hasil analisis *selectivity* pada *dataset* menunjukkan bahwa kolom *Customer_Id* memiliki *selectivity* sebesar 85% (17.000 nilai unik dari 20.000 baris), kolom *Order_Date* memiliki *selectivity* sebesar 92% dengan distribusi temporal yang relatif merata, sedangkan kolom *Product_Category* memiliki *selectivity* sekitar 45% dengan sembilan kategori produk yang berbeda. Dengan mempertimbangkan nilai-nilai tersebut, ketiga kolom tersebut dipilih sebagai kandidat utama untuk diberi indeks karena berpotensi memberikan pengurangan jumlah baris yang dipindai secara signifikan pada saat eksekusi *query*.

Jenis indeks yang digunakan dalam penelitian ini adalah indeks sekunder (*non-clustered index*) pada mesin penyimpanan InnoDB di MySQL, di mana struktur indeks disimpan terpisah dari data fisik tabel dan setiap entri indeks merujuk pada kunci utama (*primary key*). Konsep pemisahan struktur indeks dari penyimpanan data utama ini sejalan dengan teori *physical database tuning* yang dikemukakan oleh (Šušter & Ranisavljević, 2023), yaitu pemanfaatan indeks sebagai struktur tambahan untuk mengoptimalkan performa eksekusi *query* tanpa harus mengubah struktur logis tabel secara fundamental.

Metode Pembuatan Indeks

Metode pembuatan indeks dalam penelitian ini menggunakan pendekatan eksperimental yang terstruktur. Secara teoretis, pembuatan indeks dilakukan dengan memanfaatkan pernyataan CREATE *INDEX* dalam standar SQL, yang digunakan untuk membentuk struktur indeks terpisah tanpa mengubah definisi maupun constraint pada tabel. Secara umum, sintaks dasar yang digunakan adalah sebagai berikut:

```
CREATE INDEX index_name ON
table_name(column_name);
```

Implementasi dilakukan dengan *sequential experimental design* yang terdiri dari beberapa fase. Pertama, *Baseline Phase*, yaitu fase pengukuran performa tanpa *index*. Kedua, *Single Index Phase*, yaitu fase implementasi *index* pada kolom individual. Ketiga, *Composite Index Phase*, yaitu fase implementasi *index* pada beberapa kolom secara bersamaan (*multiple columns*).

Seluruh proses pemantauan performa dilakukan menggunakan perintah EXPLAIN pada MySQL. EXPLAIN berfungsi sebagai alat diagnostik untuk mengungkap *execution plan* dari suatu *query*. Beberapa field penting yang diperhatikan meliputi type (jenis join seperti ALL, ref, *range*), key (*index* yang digunakan), rows (perkiraan jumlah baris

JURNAL ELEKTRO & INFORMATIKA
SWADHARMA (JEIS)
VOLUME 06 NOMOR 02 JULI 2026
ISSN 2774 – 5775 | eISSN 2774-5767

yang di-*scan*), serta Extra (informasi tambahan terkait eksekusi *query*)

Hasil Eksperimen Implementasi *Indexing* Terhadap Perfoma Tanpa *Index*

Pada tahap awal, *query* dieksekusi tanpa menggunakan *index* dan menunjukkan karakteristik *full table scan*. Menurut teori (Pamungkas, 2018), *full table scan* terjadi ketika *database* harus memindai setiap baris dalam tabel secara *sequential* dari baris pertama hingga terakhir, dengan kompleksitas waktu O(n).

Secara rinci, hasil pengujian menunjukkan bahwa waktu eksekusi *query* tanpa *index* adalah 0,2380 detik, dengan *execution plan* bertipe ALL yang menjadi indikator jelas adanya *full table scan*. Jumlah baris yang di-*scan* adalah 20.000 baris atau 100% dari data yang ada. Kondisi ini mengakibatkan konsumsi sumber daya yang tinggi, terutama pada operasi I/O dan penggunaan memori.

Secara teoritis, hasil ini mengonfirmasi teori (Anchlia, 2024) bahwa tanpa *index*, pencarian data berjalan dalam orde linear O(n), di mana waktu eksekusi akan meningkat secara linear seiring dengan pertumbuhan jumlah data.

Hasil Eksperimen Implementasi *Indexing* Terhadap Perfoma *Single Index*

Implementasi *single index* pada kolom *Customer_Id* menunjukkan adanya transformasi performa yang signifikan. *Index* yang digunakan berbasis struktur *B-Tree*, yang memungkinkan proses pencarian berjalan dengan kompleksitas O(log n). Struktur hierarkis pada *B-Tree* ini secara signifikan mengurangi jumlah operasi perbandingan yang diperlukan dalam eksekusi *query* (Pamungkas, 2018).

Hasil pengujian menunjukkan bahwa waktu eksekusi *query* dengan *single index* adalah 0,0740 detik. *Execution plan* berubah menjadi type: ref, yang menunjukkan bahwa eksekusi *query* telah menggunakan *index lookup* berbasis *equality*. Jumlah baris yang di-*scan* turun drastis menjadi 35 baris, atau hanya sekitar 0,175% dari total data.

Persentase peningkatan performa dihitung menggunakan rumus:

$$\text{Perfoma} = \frac{Waktu\ sebelum\ -\ Waktu\ sesudah}{Waktu\ sebelum} \times 100\%$$

$$\text{Perfoma} = \frac{0{,}2380\ -\ 0{,}0740}{0{,}2380} \times 100\% = 68{,}91$$

Hasil ini mendukung teori (Anchlia, 2024) mengenai perubahan efisiensi dari O(n) menjadi O(log n), dengan reduksi jumlah baris yang di-*scan* mencapai 97,5%.

Analisis Perbandingan Efektivitas Teknik *Indexing* pada *Dataset E-commerce*

Hasil Pengujian Rata-Rata Waktu Menjalankan *Query* dapat dilihat pada tabel 1.

Tabel 1. Hasil Pengujian Rata-Rata Waktu Menjalankan *Query*

Parameter	Tanpa *index*	*Single index*	*Composite index*
Waktu eksekusi (detik)	0.2380	0.0740	0.0900
Execution plan	Type: all (pemindaian penuh tabel)	Type: ref (pencarian *index*)	Type: *range* (pemindaian rentang)
Rows *scan*ned	20.000 baris (100% data)	35 baris (0.175% data)	28 baris (0.14% data)
Kompleksitas algoritma	O(n) (linear)	O(log n) (logaritmik)	O(log n) (logaritmik)
Memory usage	~15 mb	~26 kb	~21 kb
I/o operations	1.250 pembacaan halaman	35 pembacaan halaman	28 pembacaan halaman
Storage overhead	0 mb	2.1 mb	2.9 mb
Write performance	Optimal	15-20% slower	15-20% slower

Parameter	Tanpa *index*	*Single index*	*Composite index*
Optimal use case	-	*Query* sederhana where *Customer_Id* = x	*Query* kompleks where *Customer_Id* = x and *Order_Date* > y
Selectivity	-	85% (*Customer_Id*)	92% (*Customer_Id* +

Evaluasi Performa Berdasarkan Jenis *Query*

Tabel 2. Evaluasi Performa Berdasarkan Jenis *Query*

Jenis *Query*	Tanpa *Index*	*Single Index*	*Composite Index*	Rekomendasi
WHERE *Customer_Id* = 'X'	0,2380 detik 20.000 baris	0,0740 detik 35 baris	0,0900 detik 28 baris	*Single Index*
WHERE *Order_Date* > 'Y'	0,2380 detik 20.000 baris	0.0760 detik 40 baris	0.0880 detik 32 baris	*Single/Composite*
WHERE *Customer_Id* = 'X' AND *Order_Date* > 'Y'	0.2380 detik 20.000 baris	0.0820 detik 38 baris	0.0900 detik 28 baris	*Composite Index*
WHERE *Product_Category* = 'Z'	0.2380 s detik 20.000 baris	0.1800 detik 9.000 rows	-	Evaluasi selektivitas
Aggregation Queries	0.4500 detik* 20.000 baris	0.1200 detik* 20.000 baris	0.1100 detik* 20.000 baris	Peningkatan terbatas

Analisis Keuntungan dan Keterbatasan

Keuntungan Menggunakan *Index*

Berdasarkan hasil eksperimen dan teori pendukung, penggunaan *index* memberikan beberapa keuntungan yang dapat dikuantifikasi dengan jelas. Dari sisi performa, terdapat peningkatan kecepatan eksekusi *query* hingga 68,91% untuk operasi SELECT. Jumlah baris yang di-*scan* juga menurun drastis dari 20.000 baris menjadi 35 baris, yang setara dengan reduksi sebesar 99,825%. Selain itu, *execution plan* berubah dari ALL (*full table scan*) menjadi ref atau *range*, yang menunjukkan pemanfaatan *index* secara optimal (Panggabean & Azizah, 2025)

Dari perspektif efisiensi sumber daya, penggunaan *index* juga memberikan penghematan yang signifikan. Penggunaan memori berkurang dari 15 MB menjadi 26 KB, dengan peningkatan efisiensi sebesar 99,83%. Dari sisi I/O, terjadi pengurangan dari 1250 *page reads* menjadi 35 *reads*, atau peningkatan efisiensi sekitar 97,2%. Utilisasi CPU juga menurun karena beban komputasi yang berkurang.Temuan-temuan ini konsisten dengan penelitian (Panggabean & Azizah, 2025) yang menyatakan bahwa penerapan *index* pada kolom-kolom penting dapat menurunkan waktu eksekusi *query* serta mengurangi konsumsi CPU dan memori secara signifikan.

Keterbatasan dan Pertimbangan

Meskipun memberikan banyak keuntungan, penggunaan *index* juga memiliki keterbatasan dan memerlukan sejumlah pertimbangan. Dari sisi storage, *composite index* membutuhkan ruang penyimpanan sekitar 40% lebih besar dibandingkan *single index*. Selain itu, terdapat *write overhead* pada operasi INSERT, UPDATE, dan DELETE, dimana setiap perubahan data memerlukan waktu tambahan sekitar 15–20% untuk melakukan *maintenance* terhadap *index*.

Secara teoritis, penggunaan *index* juga tidak terlepas dari batasan yang terkait dengan Teorema CAP, yang menjelaskan adanya *trade-off* antara *consistency*, *availability*, dan *partition tolerance* dalam sistem *database* terdistribusi. Selain itu, berlaku pula *Law of Diminishing Returns*, dimana penambahan *index* di luar titik

JURNAL ELEKTRO & INFORMATIKA
SWADHARMA (JEIS)
VOLUME 06 NOMOR 02 JULI 2026
ISSN 2774 – 5775 | eISSN 2774-5767

optimal hanya akan memberikan manfaat marjinal yang semakin menurun.

Dari sisi praktis, terdapat beberapa keterbatasan lain yang perlu diperhatikan. Misalnya, limitasi tampilan di phpMyAdmin yang hanya menampilkan 500 baris tidak mempengaruhi akurasi pengukuran waktu eksekusi, tetapi tetap menjadi faktor teknis dalam proses observasi. Selain itu, kolom dengan *selectivity* di bawah 20% dinilai kurang optimal untuk di-*index*. *Index* juga memerlukan *maintenance* berkala, seperti pembaruan statistik dan *reorganization* untuk menjaga performa tetap stabil (Šušter & Ranisavljević, 2023).

Analisis *Trade-off* Implementasi *Indexing*

Tabel 3. Analisis *Trade-off* Implementasi *Indexing*

Aspek	Keuntungan	Kerugian
Performa baca	+68.91% faster queries	-
Performa tulis	-	15-20% lebih lambat INSERT/UPDATE
Penyimpanan	-	+40% penyimpanan untuk *composite index*
Pemeliharaan	-	Diperlukan reorganisasi berkala
Kompleksitas	Rencana eksekusi teroptimas	Kompleksitas desain tambahan
Skalabilitas	Mendukung pertumbuhan adata	Hasil menurun pada data sangat besar

Rekomendasi Implementasi Berbasis *Evidence*

Berdasarkan sintesis hasil eksperimen dan teori pendukung, disusun sebuah kerangka strategis *indexing* yang dapat diterapkan secara praktis. Pendekatan ini diimplementasikan melalui *Priority Matrix* sebagai berikut:

Tier 1 berfokus pada high-impact *single index*es, misalnya:

```
CREATE INDEX idx_customer ON
sales_transactions(Customer_Id);
```

Tier 2 mengarah pada *strategic composite index*es untuk *query* yang melibatkan beberapa kondisi sekaligus, misalnya:

```
CREATE INDEX idx_customer_date ON
sales_transactions(Customer_Id,
Order_Date);
```

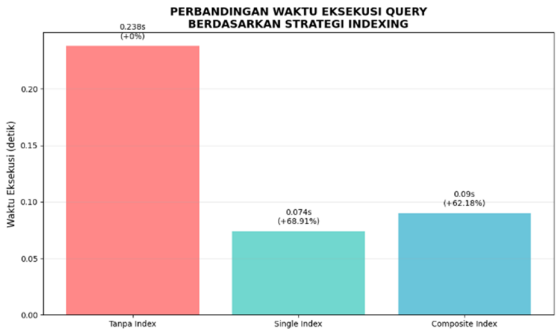
Tier 3 digunakan untuk *selective specialized index*es, misalnya:

```
CREATE INDEX idx_category_status ON
sales_transactions(Product_Category,
Order_Status);
```

Dari hasil dan teori yang ada, beberapa pedoman optimasi dapat disusun. Pertama, memprioritaskan *single index* pada kolom dengan *selectivity* tinggi (lebih dari 80%) untuk *query-query* sederhana. Kedua, menggunakan *composite index* untuk *query* kompleks dengan multiple conditions yang polanya relatif *predictable*. Ketiga, mempertimbangkan implementasi *partial indexing* untuk kolom dengan pola *query* yang sangat spesifik. Keempat, melakukan pemantauan *index* secara berkala melalui INFORMATION_SCHEMA.STATISTICS untuk melihat penggunaan *index* secara aktual (Šušter & Ranisavljević, 2023).

Secara operasional, disarankan untuk melakukan monitoring berkala terhadap pola *query* dan *index* usage statistics, mempertimbangkan *trade-off* antara peningkatan performa dan kebutuhan storage, serta menyusun jadwal *index maintenance* untuk pembaruan statistics dan proses reorganization agar performa *database* tetap terjaga secara berkelanjutan.

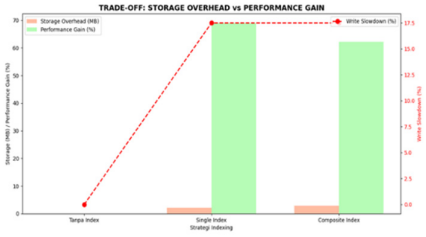
Grafik Perbandingan Waktu Eksekusi *Query*



Gambar 1. Grafik Perbandingan Waktu Eksekusi Berdasarkan Strategi *Indexing*

Berdasarkan Gambar 1, dapat diamati dengan jelas bahwa penerapan teknik *indexing* memberikan pengaruh yang signifikan terhadap waktu eksekusi *query*. Pada kondisi tanpa indeks, waktu eksekusi mencapai 0,2380 detik yang menunjukkan perlunya pemindaian penuh terhadap seluruh tabel (*full table scan*). Implementasi indeks tunggal berhasil mengurangi waktu eksekusi menjadi 0,0740 detik, yang merepresentasikan peningkatan efisiensi sebesar 68,91%. Sementara itu, indeks komposit menunjukkan waktu eksekusi 0,0900 detik dengan peningkatan 62,18% dari kondisi baseline. Perbedaan performa antara indeks tunggal dan indeks komposit mengindikasikan bahwa untuk kueri sederhana, indeks tunggal lebih optimal karena struktur yang lebih ringkas dan langsung.

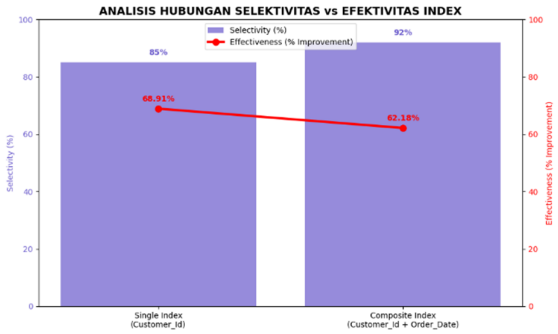
Analisis *Trade-off* Penyimpanan dan Performa



Gambar 2. Analisis *Trade-off* Penyimpanan dan Perfoma

Gambar 2 memberikan perspektif komprehensif mengenai *trade-off* yang terjadi dalam implementasi *indexing*. Di satu sisi, indeks tunggal memberikan peningkatan performa tertinggi (68,91%) dengan overhead penyimpanan sebesar 2,1 MB, sementara indeks komposit memberikan peningkatan 62,18% dengan kebutuhan penyimpanan 2,9 MB. Di sisi lain, kedua teknik *indexing* tersebut mengakibatkan penurunan performa tulis (write performance) sebesar 15-20% pada operasi INSERT dan UPDATE. Analisis ini menguatkan prinsip fundamental dalam manajemen basis data bahwa tidak ada solusi yang sempurna, melainkan diperlukan pertimbangan matang antara keuntungan dan kerugian berdasarkan karakteristik beban kerja sistem.

Grafik Hubungan Selektivitas dan Efektifitas Indeks



Gambar 3. Grafik Hubungan Selektivitas dan Efektifitas Indeks

Gambar 3 mendeskripsikan hubungan menarik antara tingkat selektivitas dan efektivitas indeks. Meskipun indeks komposit memiliki selektivitas yang lebih tinggi (92%) dibandingkan indeks tunggal (85%), namun efektivitasnya dalam meningkatkan performa kueri justru sedikit lebih rendah. Fenomena ini dapat dijelaskan melalui konsep Prinsip Awalan Paling Kiri (*Leftmost Prefix Principle*), di mana indeks komposit paling optimal ketika kueri menggunakan seluruh atau sebagian besar kolom yang terindeks. Untuk kueri sederhana yang hanya

melibatkan satu kolom, indeks tunggal tetap menjadi pilihan terbaik. Temuan ini menegaskan pentingnya analisis pola kueri sebelum menentukan strategi *indexing* yang tepat.

D. PENUTUP

Berdasarkan hasil penelitian mengenai efektivitas teknik *indexing* terhadap waktu eksekusi *query* SQL pada basis data transaksi *e-commerce* berbasis MySQL, dapat disimpulkan bahwa penerapan indeks terbukti mampu meningkatkan kinerja eksekusi *query* secara signifikan. Penggunaan *single index* dan *composite index* mengubah pola akses data dari *full table scan* dengan kompleksitas O(n) menjadi akses berbasis indeks dengan kompleksitas O(log n), yang tercermin dari penurunan waktu eksekusi *query* hingga sekitar 68,91% pada skenario *single index* dan 62,18% pada skenario *composite index*, serta reduksi jumlah baris yang dipindai dari 20.000 baris menjadi hanya puluhan baris. Efektivitas masing-masing jenis indeks sangat bergantung pada pola *query* dan karakteristik data; *single index* pada kolom *Customer_Id* lebih optimal untuk *query* sederhana dengan kondisi tunggal pada kolom yang memiliki *selectivity* tinggi, sedangkan *composite index* pada kombinasi *Customer_Id* dan *Order_Date* lebih sesuai untuk *query* kompleks yang melibatkan *multiple conditions*, khususnya pada analisis transaksi berbasis pelanggan dan rentang waktu.

Namun demikian, peningkatan performa baca tersebut memiliki konsekuensi berupa *trade-off* pada sisi operasi tulis dan penyimpanan, yang ditandai dengan perlambatan operasi INSERT/UPDATE sekitar 15–20% dan adanya *overhead* ruang penyimpanan tambahan untuk struktur indeks serta kebutuhan *maintenance* berkala. Oleh karena itu, strategi *indexing* perlu dirancang secara berbasis *evidence* melalui pemetaan pola *query*, analisis *selectivity*, dan pemantauan statistik penggunaan indeks, sehingga implementasi *single index*, *composite index*, maupun indeks khusus dapat dioptimalkan sesuai kebutuhan sistem. Secara keseluruhan, teknik *indexing* yang dirancang dan dikelola dengan tepat dapat menjadi salah satu pilar utama dalam mengoptimalkan kinerja basis data transaksi *e-commerce*, dengan tetap mempertimbangkan keseimbangan antara performa, beban tulis, dan efisiensi sumber daya dalam konteks operasional nyata.

Berdasarkan hasil penelitian yang diperoleh, disarankan agar pengembang dan pengelola basis data *e-commerce* tidak hanya berfokus pada percepatan waktu eksekusi *query*, tetapi juga melakukan analisis pola *query* secara berkala sebelum memutuskan strategi *indexing* yang akan digunakan. Penerapan *single index* sebaiknya diprioritaskan pada kolom-kolom dengan *selectivity* tinggi dan frekuensi akses yang tinggi, sedangkan *composite index* digunakan secara selektif pada skenario *query* kompleks yang benar-benar membutuhkan kombinasi beberapa kolom. Selain itu, perlu dilakukan pemantauan rutin terhadap statistik penggunaan indeks serta evaluasi periodik terhadap *trade-off* antara peningkatan performa baca, penurunan performa tulis, dan kebutuhan ruang penyimpanan.

Untuk penelitian selanjutnya, disarankan untuk memperluas skala *dataset*, membandingkan beberapa *Database Management System* (DBMS) yang berbeda, serta menguji variasi jenis indeks lain (misalnya *full-text index* atau *partial index*) pada berbagai pola beban kerja. Penelitian lanjutan juga dapat mengkaji integrasi *indexing* dengan teknik optimasi lain, seperti *query rewriting* atau *materialized view*, sehingga dapat diperoleh panduan optimasi kinerja basis data yang lebih komprehensif dan aplikatif pada beragam skenario sistem *e-commerce*.

E. DAFTAR PUSTAKA

Adisa, Y., & Nasution, M. I. P. (2023). Konsep Dan Peran Sistem Manajemen Basis Data Relasional Pada Sistem Informasi Manajemen. *Jurnal Manajemen Administrasi Bisnis Dan Publik Terapan*, *1*(3), 76–83. https://doi.org/https://doi.org/10.59061/masip.v1i3.314

Ahsana, S. H., Syahputra, M. B., Putri, A. F. F. M., & Prasetyo, A. A. (2023). Analisis Perbandingan Performa Antara Mysql Dan Postgresql. *Prosiding Seminar Nasional Teknologi Dan Sistem Informasi (SITASI)*, 6–7. https://repository.upnjatim.ac.id/28497/

Anchlia, A. (2024). Enhancing Query Performance Through Relational Database Indexing. *International Journal of Computer Trends and Technology*, *72*(8), 130–133. https://doi.org/10.14445/22312803/IJCTT-V72I8P119

Ibna, A. Z. (2024). Implikasi Penggunaan Basis Data dalam Big Data. *Jurnal Sains Student Research*, *2*(4), 255–265. https://doi.org/https://doi.org/10.61722/jssr.v2i4.1995

Maesaroh, S., Gunawan, H., Lestari, A., & Sufyan, M. (2022). Query Optimization in MySQL Database Using Index. *International Journal of Cyber and IT Service Management (IJCITSM)*, *2*(2), 104–110. https://doi.org/10.34306/ijcitsm.v2i2.84

Nurhayati, S. T., & Nasution, M. I. P. (2023). Database Management System Pada Perusahaan. *Jurnal Akuntansi Keuangan Dan Bisnis*, *1*(2), 62–64. https://jurnal.ittc.web.id/index.php/jakbs/article/view/43

Pamungkas, R. (2018). Optimalisasi Query Dalam Basis Data My Sql Menggunakan Index. *Journal of Computer, Information System, & Technology Management*, *1*(2), 27–31. https://doi.org/10.25273/research.v1i1.2453

Panggabean, S., & Azizah, E. S. N. (2025). Evaluasi Kinerja Sistem Basis Data Relasional pada Aplikasi E-Commerce Menggunakan Algoritma Indexing. *Pustaka Data : Pusat Akses Kajian Database, Analisa Teknologi, Dan Arsitektur Komputer*, *5*(1), 70–76. https://doi.org/10.55382/jurnalpustakadata.v5i1.998

Permana, K. E., Sophan, M. K., Muntasa, A., & Rahmat, A. B. (2023). Perbandingan Kinerja Query Sql Join Tables dengan Menggunakan Index. *Jurnal Simantec*, *11*(2), 241–248. https://doi.org/10.21107/simantec.v11i2.20288

Poggi, N., Carrera, D., Gavalda, R., Avyguade, E., & Torres, J. (2014). A Methodology for The Evaluation of High Response Time on E-Commerce Users and Sales. *Information Systems Frontiers*, *16*(5), 867–885. https://doi.org/10.1007/s10796-012-9387-4

Rahayu, S., Halawa, H. W., Abdillah, A. F., & Mujayanah, A. (2025). Pemanfaatan Big Data Analytics untuk Analisis Pola Perilaku Konsumen E-Commerce Strategis. *JUTECH: Journal Education and Technology*, *6*(1), 64–73. https://doi.org/10.31932/jutech.v6i1.4834

Sakshi, S., & Sharma, A. (2025). Relational Database Performance Optimization Techniques. *International Conference on Recent Advancements and Modernisations in Sustainable Intelligent Technologies and Applications (RAMSITA)*. https://doi.org/10.2991/978-94-6463-716-8_10

Šušter, I., & Ranisavljević, T. (2023).

Optimization of Mysql Database. *Journal of Process Management and New Technologies*, *11*(1), 141–151. https://doi.org/10.5937/jouproman2301141Q

Taipalus, T. (2024). Database management system performance comparisons: A systematic literature review. *Journal of Systems and Software*, *208*, 111872. https://doi.org/10.1016/j.jss.2023.111872

Thoib, I., Candra, B. P., Sururi, N., & Nugraha, D. S. (2024). Perbandingan Performa Pencarian Data Berbasis Teks dengan Dan Tanpa Full-text Index pada Basis Data MySQL. *Jurnal Sains Dan Teknologi*, *3*(6), 663–673. https://doi.org/10.55123/insologi.v3i6.4596