Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

QUERY OPTIMIZATION ON SEMI STRUCTURED MYSQL RETAIL DATA USING GENERATED COLUMNS: A COMPARATIVE STUDY

Adi Handika[1], Muqtafiy Muhammad[2], Muhammad Azka Bani Shalih[3], Nabil Yudha Syahputra[4], Imam Prayogo Pujiono[5]

[1234]Program Studi Bisnis Digital, UIN K.H. Abdurrahman Wahid, [5]Program Studi Informatika, UIN K.H. Abdurrahman Wahid

Jalan Pahlawan Km. 5, Rowolaku, Kecamatan Kajen, Kabupaten Pekalongan, Jawa Tengah 51161.
[1]adi.handika25004@mhs.uingusdur.ac.id, [2]muqtafiy.muhammad25047@mhs.uingusdur.ac.id,
[3]muhammad.azka.bani25055@mhs.uingusdur.ac.id, [4]nabil.yudha.syahputra25084@mhs.uingusdur.ac.id,
[5]imam.prayogopujiono@uingusdur.ac.id

Abstract

Query processing on JSON data stored in MySQL without adequate optimization leads to significant performance degradation, particularly when the JSON_EXTRACT function is repeatedly applied across large-scale datasets without index support. This study proposes the use of generated columns as an optimization strategy to address this limitation. The study evaluates query execution performance across three storage models in MySQL: the relational model (retail_relational), the unoptimized JSON model (retail_json_raw), and the JSON model with generated columns and indexes (retail_json_gc), using a public retail transaction dataset consisting of 1,048,575 records and eight attributes. Four query scenarios were examined, including equality filtering by country and stock code, range filtering by price, and GROUP BY aggregation by country. Each query was executed ten times and analyzed using EXPLAIN to evaluate execution plans and index utilization. Statistical validation was conducted using the Kruskal Wallis and Mann Whitney U tests with a significance level of $\alpha = 0.05$. The results demonstrate that the JSON_GC model achieved the highest performance across all scenarios. In the stock code equality query, JSON_GC achieved an average execution time of 0.006 seconds compared to 1.577 seconds for the relational model and 1.950 seconds for JSON_Raw, representing improvements of 263-fold and 325-fold respectively. In the GROUP BY aggregation query, JSON_GC required only 0.002 seconds compared to over 1.6 seconds for both alternative models. Statistical testing confirmed that all performance differences between models were significant ($p < 0.05$). These findings indicate that generated columns effectively enable B-Tree index scans and covering indexes, thereby eliminating costly full table scans and significantly improving query efficiency for semi-structured JSON-based data management in MySQL.

Keywords : *MySQL 8.0, Semi Structured Data, Generated Column, B-Tree Index, Query Performance*

Abstrak

Pemrosesan query pada data JSON yang disimpan di MySQL tanpa optimasi yang memadai menyebabkan penurunan performa yang signifikan, terutama ketika fungsi JSON_EXTRACT diterapkan secara berulang pada dataset berskala besar tanpa dukungan indeks. Penelitian ini mengusulkan penggunaan generated columns sebagai strategi optimasi untuk mengatasi permasalahan tersebut. Penelitian mengevaluasi performa eksekusi query pada tiga model penyimpanan data di MySQL, yaitu model relasional (retail_relational), model JSON tanpa optimasi (retail_json_raw), dan model JSON dengan generated columns dan indeks (retail_json_gc), menggunakan dataset transaksi retail publik yang terdiri

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

dari 1.048.575 record dengan delapan atribut. Empat skenario query diuji, meliputi filter kesamaan nilai berdasarkan negara dan kode barang, filter rentang harga, serta agregasi GROUP BY berdasarkan negara. Setiap query dijalankan sebanyak sepuluh kali dan dianalisis menggunakan EXPLAIN untuk mengevaluasi rencana eksekusi dan pemanfaatan indeks. Validasi statistik dilakukan menggunakan uji Kruskal Wallis dan Mann Whitney U dengan tingkat signifikansi $\alpha = 0,05$. Hasil penelitian menunjukkan bahwa model JSON_GC menghasilkan performa terbaik pada seluruh skenario pengujian. Pada query filter kode barang, JSON_GC memperoleh rata-rata waktu eksekusi sebesar 0,006 detik dibandingkan 1,577 detik pada model relasional dan 1,950 detik pada JSON_Raw, atau masing-masing 263 kali dan 325 kali lebih cepat. Pada query agregasi GROUP BY, JSON_GC hanya memerlukan 0,002 detik dibandingkan lebih dari 1,6 detik pada kedua model lainnya. Hasil pengujian statistik menunjukkan bahwa seluruh perbedaan performa antar model signifikan secara statistik ($p < 0,05$). Temuan ini menunjukkan bahwa generated columns mampu mengaktifkan pemanfaatan indeks B-Tree dan covering index secara efektif, sehingga dapat menghilangkan full table scan yang mahal dan meningkatkan efisiensi query secara signifikan pada pengelolaan data semi terstruktur berbasis JSON di MySQL.

Kata kunci : *MySQL 8.0, Data Semi Terstruktur, Generated Column, Indeks B-Tree, Performa Kueri*

1. PENDAHULUAN

Perkembangan sistem informasi telah mendorong perubahan karakteristik data dari yang semula dominan terstruktur menjadi semakin beragam, termasuk data semi terstruktur yang lebih fleksibel dalam merepresentasikan atribut yang berubah ubah [1]. Dalam konteks ini, basis data relasional tidak lagi hanya digunakan untuk menyimpan data tabular konvensional, tetapi juga mulai mengakomodasi format seperti JSON agar mampu mendukung kebutuhan aplikasi modern [2]. MySQL sebagai salah satu sistem manajemen basis data yang banyak digunakan telah menyediakan dukungan terhadap penyimpanan dan pemrosesan data JSON sejak versi 5.7.8, sehingga memungkinkan integrasi antara model relasional dan semi terstruktur dalam satu lingkungan pengelolaan data.

Meskipun demikian, fleksibilitas tersebut tidak selalu diikuti oleh efisiensi, karena pengambilan data langsung dari atribut JSON cenderung menambah beban pemrosesan, khususnya ketika query dijalankan pada data berukuran besar atau pada pola akses yang berulang. Kondisi ini menunjukkan bahwa pemanfaatan JSON di MySQL memerlukan strategi optimasi yang tepat agar keuntungan fleksibilitas struktur tidak dibayar dengan penurunan performa query yang signifikan. Setiap query pada kolom JSON murni membutuhkan ekstraksi nilai menggunakan fungsi JSON_EXTRACT secara berulang pada saat eksekusi, yang secara langsung meningkatkan overhead komputasi dan waktu respons sistem. Kondisi ini telah diidentifikasi dalam beberapa penelitian terdahulu, namun solusi berbasis fitur native MySQL seperti generated columns belum menjadi fokus kajian yang terpadu [3], [4].

Salah satu fitur yang dapat digunakan untuk mengatasi persoalan tersebut adalah generated columns, yaitu mekanisme yang memungkinkan nilai tertentu dari atribut JSON diturunkan menjadi kolom turunan yang dapat diakses secara lebih efisien oleh sistem query [5]. Pendekatan ini memungkinkan sebagian proses ekstraksi dipindahkan ke tahap definisi skema, sehingga akses terhadap elemen elemen JSON yang sering digunakan dalam klausa pencarian, pengelompokan, maupun agregasi dapat dilakukan dengan overhead yang lebih rendah melalui dukungan indeks B Tree reguler [3], [6]. Studi-studi terdahulu tentang optimasi query pada MySQL umumnya berfokus pada optimasi relasional konvensional, indexing umum, atau perbandingan antarsistem basis data, sehingga belum ada yang secara khusus mengukur dampak generated columns terhadap performa query pada data semi terstruktur berbasis JSON berskala jutaan record dalam satu lingkungan MySQL yang terkontrol [7], [8].

Penelitian ini menggunakan dataset transaksi retail publik (Online Retail II) yang memuat informasi invoice, kode barang, deskripsi produk, jumlah pembelian, tanggal transaksi, harga, identitas pelanggan, dan negara dengan total lebih dari satu juta record. Dataset tersebut ditransformasikan sebagian atributnya ke dalam

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

format JSON untuk mensimulasikan kebutuhan penyimpanan semi terstruktur dalam lingkungan MySQL yang terkontrol [9]. Dengan rancangan demikian, perbedaan performa query yang muncul dapat dianalisis secara objektif sebagai akibat dari pemilihan skema data, bukan karena perbedaan karakter dataset. Rancangan ini secara langsung menjawab gap yang belum dijawab oleh penelitian sebelumnya, yaitu perbandingan empiris ketiga model penyimpanan dalam satu lingkungan MySQL yang terkontrol Berdasarkan latar belakang tersebut, penelitian ini berfokus pada pertanyaan mengenai bagaimana perbedaan performa query antara model relasional dan model semi terstruktur berbasis JSON, serta sejauh mana generated columns mampu meningkatkan efisiensi eksekusi query.

Tujuan penelitian ini adalah menganalisis perbedaan performa query antara model data relasional dan model data semi terstruktur berbasis JSON pada MySQL, mengevaluasi pengaruh penggunaan generated columns terhadap efisiensi eksekusi query, serta mengidentifikasi pola query yang paling responsif terhadap penerapan teknik tersebut. Penelitian ini diharapkan dapat memberikan kontribusi praktis bagi pengembang sistem dan pengelola basis data dalam menentukan pendekatan penyimpanan yang sesuai antara kebutuhan fleksibilitas struktur dan tuntutan performa akses pada dataset skala besar. Kontribusi utama penelitian ini terletak pada evaluasi empiris generated columns pada data retail semi terstruktur berskala lebih dari satu juta record menggunakan kombinasi execution plan analysis, query selectivity analysis, dan statistical significance testing non parametrik (Kruskal Wallis dan Mann Whitney U) dalam lingkungan MySQL 8.0 berbasis consumer grade hardware, yang belum pernah dilakukan secara terpadu pada konteks dataset e commerce publik sebelumnya [7], [10], [11].

2. TINJAUAN PUSTAKA

2.1. Basis Data Relasional dan Keterbatasannya pada Data Semi Terrstruktur

Model basis data relasional yang diperkenalkan oleh Codd telah menjadi fondasi penyimpanan data selama lebih dari lima dekade. Model ini mengorganisasikan data dalam bentuk tabel dengan skema yang kaku dan relasi antar tabel melalui kunci asing. Optimasi query pada model relasional sangat bergantung pada ketersediaan indeks B Tree yang memungkinkan akses data secara efisien tanpa full table scan [12]. Namun, data semi terstruktur yang tidak memiliki skema tetap sulit diakomodasi secara efisien dalam model relasional [13]. Survei komprehensif oleh Yuan et al. menunjukkan bahwa kebutuhan untuk menyimpan dan memproses data semi terstruktur dalam basis data relasional terus meningkat seiring dominasi RDBMS di pasar, sehingga mendorong berbagai pendekatan pemetaan skema yang adaptif untuk mengakomodasi data JSON dan graf ke dalam struktur relasional [14]. Variasi struktur pada data JSON dari berbagai sumber NoSQL justru menjadi tantangan tersendiri ketika harus diintegrasikan ke dalam lingkungan relasional [2]. Hal ini mendorong penelitian lebih lanjut mengenai strategi hybrid yang menggabungkan keduanya.

2.2. Perbandingan SQL dan NoSQL pada Beban Kerja *E Commerce*

Basis data NoSQL ke dalam empat kategori utama (dokumen, kolom, kunci nilai, dan graf) serta membandingkannya dengan basis data relasional dari sisi fleksibilitas skema, skalabilitas horizontal, dan konsistensi transaksi [10], [13]. Penelitian tersebut menyimpulkan bahwa basis data dokumen seperti MongoDB menawarkan fleksibilitas tinggi untuk data tidak beraturan, namun mengalami penurunan performa pada operasi agregasi kompleks dibandingkan SQL. Sistem basis data SQL dan NoSQL pada sistem relasional masih unggul dalam konsistensi dan kemampuan query adhoc, sementara NoSQL lebih efisien untuk beban tulis skala besar [15]. Studi terbaru yang dipublikasikan di IEEE secara khusus membandingkan model relasional, graf, kolom lebar, kunci-nilai, dan dokumen untuk beban kerja e-commerce, menyimpulkan bahwa pemilihan model basis data yang tepat sangat bergantung pada karakteristik fungsionalitas seperti pemrosesan pesanan dan manajemen hubungan pelanggan [16]. Tinjauan literatur sistematis lebih lanjut mengonfirmasi bahwa basis data SQL terbukti andal untuk query terstruktur yang kompleks dan menjaga integritas transaksi, meskipun kurang fleksibel dalam menangani data tidak terstruktur berskala besar [17]. Secara spesifik membandingkan basis data relasional dan berorientasi dokumen untuk beban kerja e commerce dengan dataset

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

transaksi nyata, dan menemukan bahwa basis data relasional dengan pengindeksan yang tepat mampu menyaingi performa NoSQL pada skenario baca yang intens [18]. Studi-studi tersebut membandingkan sistem basis data yang berbeda (SQL vs. NoSQL), namun belum secara mendalam mengeksplorasi optimasi dalam satu sistem MySQL yang sama menggunakan fitur JSON native beserta generated columns sebagai strategi hibrida.

2.3. Penyimpanan dan Pemrosesan JSON di MySQL

Dukungan tipe data JSON secara native di MySQL diperkenalkan pada versi 5.7.8, memungkinkan penyimpanan dokumen JSON dengan validasi sintaks otomatis dan akses melalui fungsi JSON_EXTRACT, JSON_UNQUOTE, dan operator path $. Performa query JSON di MySQL dan menemukan bahwa penggunaan fungsi JSON_EXTRACT pada kolom tanpa indeks mengakibatkan full table scan yang secara eksponensial meningkat seiring bertambahnya jumlah record [7]. Pemrosesan data semi terstruktur pada basis data enterprise dan menyimpulkan bahwa pendekatan hybrid yang menggabungkan penyimpanan JSON dengan kolom terindeks pada sistem relasional dapat memberikan keseimbangan antara fleksibilitas dan performa [2]. Zhang et al. mempertegas temuan ini dengan mengusulkan model konversi data JSON–SQL yang terpadu untuk lingkungan penyimpanan hybrid, dan membuktikan bahwa pendekatan kontrol berbasis JSON dapat menjembatani basis data relasional dan NoSQL secara efisien tanpa kehilangan performa yang signifikan [19].Penelitian ini menjadi landasan teoretis bagi eksplorasi generated columns sebagai mekanisme optimasi di MySQL 8.0. Lebih lanjut, strategi penyimpanan JSON pada basis data relasional untuk aplikasi e commerce bervolume tinggi dan mengonfirmasi bahwa pendekatan indeks berbasis kolom turunan secara konsisten mengungguli penyimpanan JSON murni pada beban query baca [10], [18]. Strategi pengindeksan pada basis data hybrid relasional JSON memberikan peningkatan performa signifikan yang sejalan dengan temuan penelitian ini [8].

2.4. Generated Columns dan Functional Index sebagai Strategi Optimasi

Generated columns pada MySQL memungkinkan definisi kolom turunan yang nilainya dihitung secara otomatis dari ekspresi tertentu, termasuk dari path JSON. Analisis mendalam tentang teknik B Tree modern menjelaskan bahwa indeks B Tree pada generated column bekerja identik dengan indeks pada kolom biasa, sehingga optimizer dapat memanfaatkannya untuk strategi akses ref, range, dan covering index.n dalam Star Schema Benchmark mendemonstrasikan bahwa pemilihan strategi pengindeksan yang tepat dapat memberikan perbedaan performa hingga beberapa orde magnitude pada dataset skala jutaan baris [7]. Pada MySQL 8.0, functional index hadir sebagai alternatif generated columns yang memberikan fleksibilitas serupa tanpa mendefinisikan kolom eksplisit. Belum ada studi yang secara empiris mengukur perbedaan performa antara model relasional murni, JSON tanpa optimasi, dan JSON dengan generated columns pada dataset transaksi e commerce berskala lebih dari satu juta record dalam lingkungan MySQL 8.0 yang terkontrol dengan perangkat keras kelas konsumen. Studi perbandingan MySQL, MongoDB, dll dan survei teknik optimasi query sekalipun belum menyentuh skenario generated column native di MySQL secara terfokus [20]. Studi benchmark terbaru untuk beban kerja e-commerce pada document store memperkuat pentingnya pemilihan strategi pengindeksan dan desain skema yang tepat guna meminimalkan trade-off antara performa baca dan fleksibilitas penyimpanan [10]. Penelitian ini mengisi gap tersebut dengan eksperimen kuantitatif menggunakan dataset Online Retail II [15] pada spesifikasi AMD Ryzen 5 7430U, RAM 8 GB, dan SSD 256 GB, dilengkapi dengan pengujian statistik non parametrik yang lebih robust.

3. METODOLOGI PENELITIAN

3.1. Tahapan Penelitian

Penelitian ini menggunakan pendekatan kuantitatif eksperimental untuk mengevaluasi performa query pada tiga model penyimpanan data di MySQL. Variabel independen dalam penelitian ini adalah model penyimpanan data yang terdiri atas tiga skenario, yaitu model relasional (retail_relational) sebagai kondisi dasar (baseline), model JSON tanpa optimasi (retail_json_raw) yang merepresentasikan

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

kondisi awal adopsi JSON tanpa strategi pengindeksan, dan model JSON dengan generated columns (retail_json_gc) sebagai pendekatan yang menggabungkan fleksibilitas JSON dengan efisiensi akses berbasis indeks. Ketiga model ini dipilih karena merepresentasikan spektrum pendekatan yang umum dijumpai dalam praktik pengembangan sistem berbasis MySQL, sehingga perbandingannya memungkinkan evaluasi yang komprehensif terhadap trade-off antara fleksibilitas skema dan performa query. Variabel dependennya adalah performa query yang diukur melalui waktu eksekusi rata rata, nilai minimum dan maksimum, serta karakteristik rencana eksekusi query berupa tipe akses, penggunaan indeks, dan jumlah baris yang dipindai. Desain penelitian dilakukan secara bertahap dengan menjadikan tabel relasional sebagai kondisi dasar, kemudian membandingkannya dengan kedua model JSON. Validasi perbedaan performa antar model dilakukan menggunakan uji Kruskal Wallis untuk mendeteksi perbedaan signifikan antara ketiga model secara keseluruhan, dilanjutkan dengan uji Mann Whitney U sebagai uji lanjut (post-hoc) untuk membandingkan setiap pasangan model secara langsung. Kedua uji non-parametrik ini dipilih karena data waktu eksekusi pada lingkungan pengujian nyata tidak dapat diasumsikan berdistribusi normal, mengingat adanya potensi outlier akibat fluktuasi sistem operasi dan aktivitas background process, dengan tingkat signifikansi yang ditetapkan sebesar $\alpha = 0,05$.

3.2. Dataset dan Lingkungan Pengujian

Dataset Online Retail II diperoleh dari UCI Machine Learning Repository https://archive.ics.uci.edu/dataset/502/online+retail+ii dalam format .xlsx dan diekstrak menggunakan library pandas pada Python 3. Dataset tersusun dalam dua lembar data periode 2009-2010 dan 2010-2011, kemudian digabungkan menjadi satu himpunan data untuk keperluan eksperimen, sehingga memuat total 1.048.575 baris data dengan delapan atribut. Pemilihan dataset ini didasarkan pada tiga pertimbangan: (1) bersifat publik dan dapat diakses bebas sehingga penelitian dapat direplikasi, (2) memiliki skala lebih dari satu juta record yang representatif untuk menguji performa pada beban data besar, serta (3) memiliki atribut yang beragam sehingga dapat mensimulasikan berbagai jenis operasi query yang umum dalam sistem e-commerce. Sebelum proses pengujian dilakukan, data dibersihkan dari nilai kosong pada atribut Description (4.372 nilai kosong) dan Customer ID (236.682 nilai kosong atau 22,6% dari total data), inkonsistensi format tanggal, serta transaksi dengan nilai Quantity atau Price negatif yang merepresentasikan retur. Data yang telah bersih kemudian diimpor ke MySQL 8.0 menggunakan perintah LOAD DATA INFILE. Lingkungan pengujian menggunakan MySQL 8.0 Workbench yang dijalankan pada laptop dengan prosesor AMD Ryzen 5 7430U, penyimpanan SSD 256 GB, dan memori RAM 8 GB. Spesifikasi ini merepresentasikan kelas perangkat keras konsumen yang umum digunakan oleh pengembang dan peneliti di lingkungan akademik.

Tabel 1. Struktur Atribut Dan Statistik Dataset Transaksi Retail

No	Atribut	Keterangan
1	Invoice (String)	Nomor faktur transaksi.
2	StockCode (String)	Kode produk unik per item.
3	Description (String)	Deskripsi Produk (4.372 nilai kosong).
4	Quantity (Integer)	Jumlah unit (min: -74.215, max: 74.215).
5	InvoiceDate (DateTime)	Waktu transaksi (01/12/2009 – 04/12/2011
6	Price (Float)	Harga per unit dalam GBP (rata rata: 4,63).
7	CustomerID (Float)	ID pelanggan (22,6% nilai kosong).
8	Country (String)	Negara transaksi (43 negara, UK dominan 91,9%).

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

3.3. Konstruksi Tiga Model Penyimpanan Data

Data yang sama dimuat ke dalam tiga tabel MySQL dengan struktur berbeda untuk memastikan perbandingan yang adil. Pertama, tabel retail_relational menyimpan seluruh delapan atribut dalam kolom-kolom terpisah dengan tipe data yang sesuai, tanpa indeks tambahan pada kolom country, stock_code, maupun price. Kedua, tabel retail_json_raw menyimpan atribut atribut transaksi (kecuali primary key) ke dalam satu kolom bertipe LONGTEXT bernama item_detail dalam format JSON, tanpa indeks pada nilai nilai di dalamnya. Ketiga, tabel retail_json_gc menggunakan struktur yang sama dengan retail_json_raw, namun ditambahkan tiga generated column virtual beserta indeks B Tree, yaitu country_gc yang mengekstrak nilai country dari JSON, stock_code_gc yang mengekstrak nilai stock_code, dan price_gc yang mengekstrak nilai price dengan tipe data DECIMAL.

Pemilihan tiga model penyimpanan data dalam penelitian ini didasarkan pada pertimbangan bahwa ketiganya merepresentasikan spektrum pendekatan yang umum dijumpai dalam praktik pengembangan sistem berbasis MySQL. Model relasional dipilih sebagai kondisi dasar (baseline) karena merupakan pendekatan konvensional yang paling banyak digunakan. Model JSON tanpa optimasi (JSON_Raw) dipilih untuk merepresentasikan kondisi awal yang lazim terjadi ketika pengembang mengadopsi penyimpanan semi-terstruktur tanpa mempertimbangkan strategi pengindeksan. Model JSON dengan generated columns (JSON_GC) dipilih sebagai pendekatan yang dihipotesiskan mampu menggabungkan fleksibilitas JSON dengan efisiensi akses berbasis indeks. Dengan demikian, perbandingan ketiga model ini memungkinkan evaluasi yang komprehensif terhadap trade-off antara fleksibilitas skema dan performa query.

3.4. Skenario Query dan Parameter Evaluasi

Empat skenario query dirancang untuk mencakup jenis operasi yang paling umum dalam sistem e-commerce, meliputi filter kesetaraan dengan selektivitas rendah, filter kesetaraan dengan selektivitas tinggi, filter rentang nilai, dan agregasi pengelompokan. Kombinasi keempat skenario ini dipilih untuk mengungkap perilaku optimizer MySQL pada berbagai kondisi akses data, mulai dari broad retrieval hingga highly selective lookup dan operasi agregasi kompleks. Query 1 melakukan filter equality berdasarkan negara (country = 'United Kingdom') yang menghasilkan 963.819 record dari total 1.048.575 record atau sekitar 91,9% data. Query 2 melakukan filter equality berdasarkan kode barang (stock_code = '85123A') yang menghasilkan 5.878 record atau sekitar 0,56% data sehingga memiliki selektivitas tinggi. Query 3 melakukan filter range berdasarkan rentang harga (price BETWEEN 2.00 AND 5.00) yang menghasilkan 370.874 record. Query 4 melakukan agregasi pendapatan (SUM dari quantity dikali price) dengan pengelompokan per negara menggunakan GROUP BY yang menghasilkan 43 kelompok. Setiap query dijalankan sebanyak sepuluh kali pada masing masing skenario untuk memperoleh rata-rata yang lebih reliabel. Untuk memastikan validitas pengukuran, query cache MySQL dinonaktifkan pada setiap sesi pengujian menggunakan perintah SET SESSION query_cache_type = 0 guna mencegah hasil eksekusi pertama memengaruhi pengulangan berikutnya, dan satu run pemanasan (warm up run) dilakukan sebelum sepuluh pengulangan resmi dicatat. Waktu eksekusi diukur menggunakan performance_schema.events_statements_history pada MySQL 8.0 dengan presisi milidetik. Parameter evaluasi meliputi waktu eksekusi rata rata, nilai minimum, nilai maksimum, tipe akses pada EXPLAIN (ALL, ref, range, index), indeks yang digunakan, dan jumlah baris yang diperiksa oleh optimizer MySQL.

Tabel 2. Skenario Query Pengujian Dan Karakteristiknya

No	Kondisi Filter	Jumlah Record
Q1	country = 'United Kingdom' (Equality).	963.819 (91,9% dari total).
Q2	stock_code = '85123A' (Equality).	5.878 (0,56% dari total)
Q3	price BETWEEN 2.00 AND 5.00 (Range).	370.874 (35,4% dari total).
Q4	GROUP BY country (Agregasi).	43 kelompok negara.

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

4. HASIL DAN PEMBAHASAN

4.1. Hasil Query 1 Filter Equality Berdasarkan Country

Bagian Query 1 mengukur jumlah transaksi dari United Kingdom menggunakan tiga skenario penyimpanan yang berbeda. Pada model relasional, query dieksekusi dengan sintaks WHERE country = 'United Kingdom', sedangkan pada model JSON_Raw diperlukan fungsi JSON_UNQUOTE(JSON_EXTRACT(item_detail, '$.country')) untuk mengekstrak nilai country dari kolom JSON. Model JSON_GC cukup menggunakan kondisi WHERE country_gc = 'United Kingdom' karena nilai sudah tersedia pada generated column. Model JSON_GC menunjukkan performa terbaik dengan rata-rata 0,907 detik, lebih cepat 40,6% dibandingkan model relasional (1,503 detik) dan 56,1% lebih cepat dibandingkan JSON_Raw (2,064 detik). Analisis EXPLAIN mengonfirmasi bahwa JSON_GC menggunakan tipe akses ref pada indeks idx_country_gc, sedangkan kedua model lainnya melakukan full table scan (type: ALL) dengan memeriksa lebih dari satu juta baris. Keunggulan JSON_GC pada skenario ini tidak sebesar skenario lain karena kondisi filter menghasilkan 963.819 baris (sekitar 91,9% dari total data), sehingga manfaat selektivitas indeks berkurang [21]. Hal ini sejalan dengan prinsip yang dikemukakan oleh Saputri et al. [6] bahwa indeks B-Tree memberikan manfaat paling signifikan ketika selektivitas kondisi filter tinggi; pada kondisi low selectivity seperti Q1, biaya traversal indeks mendekati biaya full table scan sehingga selisih performa menjadi lebih kecil. Model JSON_Raw menanggung overhead tambahan dari pemanggilan fungsi JSON_UNQUOTE dan JSON_EXTRACT pada setiap baris yang diperiksa, sesuai dengan temuan Satria et al. [2] bahwa ekstraksi nilai JSON secara berulang pada saat eksekusi meningkatkan overhead komputasi secara proporsional terhadap jumlah baris yang dipindai.

4.2. Hasil Query 2 Filter Equality Berdasarkan StockCode

Bagian Query 2 merupakan skenario paling diskriminatif karena filter stock_code = '85123A' hanya menghasilkan 5.878 record dari total 1.048.575 record, atau selektivitas sebesar 0,56%. Tingkat selektivitas yang tinggi ini menjadikan efisiensi penggunaan indeks sangat berpengaruh terhadap waktu eksekusi. Hasil Query 2 menunjukkan keunggulan JSON_GC yang sangat dramatis, yaitu rata-rata 0,006 detik dibandingkan 1,577 detik pada model relasional dan 1,950 detik pada JSON_Raw, setara dengan percepatan sekitar 263 kali dibanding model relasional dan 325 kali dibanding JSON_Raw. Analisis EXPLAIN mengonfirmasi bahwa JSON_GC menggunakan kondisi Using index pada keterangan Extra, yang menandakan covering index memungkinkan MySQL memenuhi query sepenuhnya dari struktur indeks B-Tree pada kolom stock_code_gc tanpa perlu membaca data aktual dari baris tabel. Kondisi covering index ini merupakan kondisi optimal dalam eksekusi query basis data sebagaimana dijelaskan oleh Yusfa et al. [8], karena eliminasi akses ke tabel utama (table heap) secara drastis mengurangi operasi I/O disk. Temuan ini memperkuat hasil penelitian Firdaus et al. [12] yang menyimpulkan bahwa penerapan indeks pada kolom yang digunakan sebagai filter dapat memangkas waktu eksekusi hingga beberapa orde magnitude pada dataset berskala jutaan baris. Sebaliknya, model relasional yang tidak memiliki indeks pada kolom stock_code harus memindai hampir seluruh 1.048.575 baris aktual untuk menemukan 5.878 record yang relevan, sementara JSON_Raw menanggung beban serupa dengan tambahan overhead fungsi JSON_EXTRACT pada setiap baris yang diperiksa [2], [12].

4.3. Hasil Query 3 Filter Range Berdasarkan Price

Bagian Query 3 menguji kemampuan filter range dengan kondisi price BETWEEN 2.00 AND 5.00 yang menghasilkan 370.874 record. Pada model JSON_Raw, diperlukan konversi tipe data menggunakan CAST sebelum perbandingan dilakukan karena nilai price disimpan sebagai string di dalam JSON. JSON_GC kembali menunjukkan performa unggul dengan rata-rata 0,181 detik, sekitar 4,9 kali lebih cepat dari model relasional (0,889 detik) dan 7,1 kali lebih cepat dari JSON_Raw (1,283 detik). EXPLAIN menunjukkan JSON_GC menggunakan tipe akses range dengan keterangan Using where; Using index, yang menandakan indeks B-Tree pada kolom price_gc berhasil membatasi pemindaian ke subset data yang relevan. Hal ini konsisten dengan penjelasan Saputri et al. [6] bahwa indeks B-Tree mendukung operasi range scan secara efisien karena nilai-nilai tersimpan dalam urutan

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

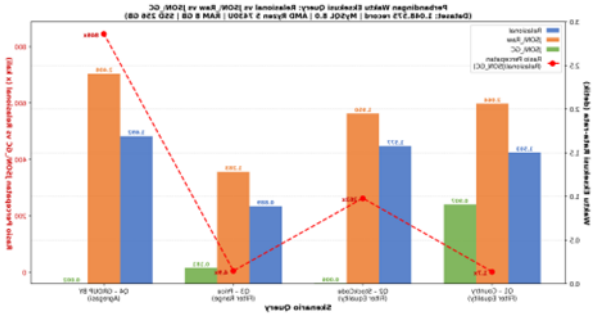
terurut, sehingga optimizer dapat langsung menentukan batas awal dan akhir pemindaian tanpa menelusuri seluruh tabel. Model relasional tidak memiliki indeks pada kolom price sehingga tetap melakukan full table scan, meskipun performa absolutnya lebih baik dari JSON_Raw karena tidak menanggung overhead fungsi CAST dan JSON_EXTRACT. Temuan ini sejalan dengan Zhang et al. [19] yang menegaskan bahwa penyimpanan nilai numerik dalam format JSON tanpa kolom turunan berindeks memaksa sistem melakukan konversi tipe data secara dinamis pada setiap baris, yang secara kumulatif menambah beban komputasi signifikan pada dataset berskala besar.

4.4. Hasil Query 4 Agregasi GROUP BY Berdasarkan Country

Bagian Query 4 menguji operasi agregasi yang menghitung total pendapatan (SUM dari perkalian quantity dan price) dengan pengelompokan berdasarkan country, menghasilkan 43 kelompok negara. Model JSON_GC menghasilkan rata-rata eksekusi hanya 0,002 detik (2 milidetik), sekitar 846 kali lebih cepat dibanding model relasional (1,692 detik) dan lebih dari 1.200 kali lebih cepat dibanding JSON_Raw (2,406 detik). Analisis EXPLAIN menunjukkan JSON_GC menggunakan tipe akses index dengan Using index, yang berarti seluruh operasi GROUP BY dapat diselesaikan melalui pemindaian indeks pada kolom country_gc tanpa perlu memuat baris data dari tabel utama. Keunggulan ini dapat dijelaskan melalui mekanisme loose index scan pada MySQL, di mana optimizer memanfaatkan sifat terurut dari indeks B-Tree untuk menyelesaikan pengelompokan secara langsung tanpa membangun tabel sementara. Sebaliknya, model relasional dan JSON_Raw keduanya menggunakan Using temporary, yang menandakan MySQL harus membangun tabel sementara di memori untuk proses pengelompokan. Van Landuyt et al. [10] dalam studi benchmark e-commerce mereka menemukan bahwa operasi agregasi merupakan titik paling kritis yang membedakan performa antar pendekatan penyimpanan, karena ketiadaan indeks yang sesuai memaksa sistem mengalokasikan memori tambahan dan melakukan sorting data secara penuh. Rasio percepatan yang mencapai 846x pada Q4 merupakan yang tertinggi dalam penelitian ini, menegaskan bahwa generated columns memberikan manfaat paling besar justru pada operasi agregasi kompleks yang lazim digunakan dalam pelaporan dan analitik sistem e-commerce.

4.5. Ringkasan Perbandingan dan Analisis EXPLAIN

Gambar 1 menyajikan grafik batang-garis yang menggabungkan perbandingan waktu eksekusi rata rata ketiga model pada keempat skenario query (batang) dengan rasio percepatan JSON_GC terhadap model relasional (garis). Grafik ini secara visual menegaskan superioritas JSON_GC yang semakin besar seiring meningkatnya selektivitas query dan kompleksitas operasi agregasi.



Gambar 1. Perbandingan Waktu Eksekusi Rata Rata (s) Dan Rasio Percepatan JSON_GC vs Relasional pada empat skenario query

Dari grafik terlihat bahwa pada Q1 (filter country dengan selektivitas rendah 91,9%), perbedaan performa antar model relatif lebih kecil dengan rasio percepatan 1,7x. Sebaliknya, pada Q2 (filter stock_code dengan selektivitas tinggi 0,56%) dan Q4 (agregasi GROUP BY), rasio percepatan JSON_GC melonjak drastis masing masing menjadi 263x dan 846x. Pola ini konsisten dengan teori bahwa manfaat indeks semakin besar ketika jumlah baris yang harus diakses lebih kecil dibandingkan total dataset, atau ketika indeks dapat mengeliminasi kebutuhan temporary table pada operasi GROUP BY [6], [8].

Perbedaan performa antar model secara fundamental bersumber dari kemampuan optimizer MySQL dalam memilih strategi akses data. Model relasional dan JSON_Raw secara konsisten mendapatkan tipe akses ALL pada EXPLAIN yang berarti full table scan, sehingga MySQL harus memeriksa lebih dari satu juta baris pada setiap eksekusi query. JSON_GC, dengan ketersediaan indeks B Tree pada generated column, memungkinkan optimizer memilih

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

strategi yang jauh lebih efisien: ref untuk filter equality, range untuk filter rentang nilai, dan index untuk pengelompokan data. Kondisi paling optimal yang dicapai adalah covering index pada Query 2, di mana MySQL dapat menyelesaikan query sepenuhnya dari struktur indeks tanpa mengakses tabel utama sama sekali [8].

Perlu dicatat bahwa model relasional dalam pengujian ini tidak dilengkapi dengan indeks pada kolom-kolom yang digunakan sebagai filter (country, stock_code, price). Hal ini dilakukan untuk memodelkan kondisi awal yang umum dijumpai dalam praktik pengembangan sistem, di mana penambahan indeks tidak selalu dilakukan sejak awal. Jika indeks ditambahkan pada kolom-kolom tersebut di model relasional, performanya akan meningkat signifikan terutama pada Query 2 dan Query 3 [8]. Namun demikian, temuan ini justru menegaskan keunggulan struktural pendekatan JSON_GC: berbeda dengan model relasional konvensional yang mengharuskan pengembang menambahkan indeks secara manual dan terpisah dari definisi skema, generated columns secara inheren mendorong pendefinisian indeks bersamaan dengan kolom turunan sejak tahap perancangan. Pola ini sejalan dengan rekomendasi Yusfa et al. [8] bahwa strategi pengindeksan yang terintegrasi dalam desain skema menghasilkan sistem yang lebih konsisten performanya dibandingkan pengindeksan yang ditambahkan secara ad-hoc pascaimplementasi.

4.6. Uji Statistik Kruskal Wallis dan Mann Whitney U

Untuk memvalidasi bahwa perbedaan waktu eksekusi antar model bukan sekadar variasi acak, dilakukan uji statistik non-parametrik Kruskal-Wallis pada setiap skenario query dengan $\alpha = 0,05$. Pemilihan uji Kruskal Wallis didasarkan pada pertimbangan teoritis bahwa data waktu eksekusi pada lingkungan pengujian nyata umumnya tidak memenuhi asumsi normalitas yang disyaratkan oleh ANOVA, karena rentan terhadap outlier akibat variasi beban sistem, cache miss, dan jitter I/O; sehingga uji non parametrik yang tidak bergantung pada asumsi distribusi menjadi pilihan yang lebih tepat dan robust. Setiap kelompok terdiri dari sepuluh pengulangan (n = 10), sehingga total observasi per skenario adalah 30 data point (3 model × 10 run). Hipotesis yang diuji adalah H_0: tidak ada perbedaan performa antar model, dan H_1: terdapat minimal satu model yang berbeda secara signifikan. Selanjutnya, uji Mann Whitney U dilakukan sebagai uji lanjut post-hoc Kruskal Wallis untuk membandingkan pasangan model secara langsung (Rel vs Raw, Rel vs GC, dan Raw vs GC), menggantikan pendekatan t-test independen yang mensyaratkan normalitas dan homogenitas varians [11], [22].

Tabel 3. Uji Kruskal Wallis (H0: Tidak Ada Perbedaan Performa Antar Model)

No	H-statistik dan p-value	Urutan Performa
Q1	25,90 dan <0,001	GC < Rel < Raw
Q2	25,88 dan <0,001	GC << Rel < Raw
Q3	26,35 dan <0,001	GC < Rel < Raw
Q4	26,18 dan <0,001	GC <<< Rel < Raw

Tabel 4. Uji Mann Whitney U Perbandingan Pasangan (Post Hoc)

Perbandingan	U-statistik	p-value
Q1: Rel vs Raw	0,0	<0,001
Q1: Rel vs GC	100,0	<0,001
Q1: Raw vs GC	100,0	<0,001
Q2: Rel vs Raw	0,0	<0,001
Q2: Rel vs GC	100,0	<0,001
Q2: Raw vs GC	100,0	<0,001
Q3: Rel vs Raw	0,0	<0,001
Q3: Rel vs GC	100,0	<0,001
Q3: Raw vs GC	100,0	<0,001
Q4: Rel vs Raw	0,0	<0,001
Q4: Rel vs GC	100,0	<0,001
Q4: Raw vs GC	100,0	<0,001

Hasil uji Kruskal Wallis pada Tabel 3 menunjukkan bahwa pada seluruh skenario query (Q1 hingga Q4), nilai H-statistik berkisar antara 25,88 hingga 26,35 dengan p-value mendekati nol ($p < 0,001$), sehingga hipotesis nol ditolak pada tingkat signifikansi 5%. Pola nilai H yang tinggi dan konsisten di seluruh skenario mencerminkan pemisahan distribusi rank yang sangat jelas antara ketiga kelompok, sejalan

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

dengan temuan Saputri et al. [6] dan Saputra et al. [7] yang menunjukkan bahwa penerapan indeks B-Tree menghasilkan kelas performa berbeda secara statistik dibandingkan tabel tanpa indeks pada dataset berskala besar. Separasi performa yang semakin ekstrem pada Q2 dan Q4, di mana JSON_GC mencapai waktu eksekusi mendekati nol, dapat dijelaskan melalui prinsip selektivitas indeks: semakin tinggi selektivitas filter, semakin besar kemampuan indeks dalam mengeliminasi full table scan [6], [8]. Pada Q4, keunggulan ini mencapai titik optimal karena indeks pada generated column tidak hanya mempercepat filter, tetapi juga mengeliminasi kebutuhan temporary table pada operasi GROUP BY yang merupakan sumber overhead dominan pada dataset berskala jutaan baris [8], [23], [24].

Uji lanjut Mann Whitney U antar pasangan model (Tabel 4) menghasilkan $p < 0{,}001$ pada seluruh kombinasi dan skenario. Nilai $U = 0{,}0$ pada perbandingan Rel vs Raw menunjukkan bahwa seluruh sepuluh run model JSON_Raw menghasilkan waktu eksekusi lebih tinggi daripada seluruh run model relasional tanpa satu pun tumpang tindih distribusi, suatu kondisi separasi sempurna yang mengindikasikan efek perbedaan yang sangat besar antar kelompok [11]. Nilai $U = 100{,}0$ pada perbandingan Rel vs GC dan Raw vs GC mengonfirmasi bahwa JSON_GC secara konsisten lebih cepat pada seluruh run tanpa pengecualian, yang secara empiris memperkuat argumentasi Yusfa et al. [8] bahwa strategi pengindeksan yang terintegrasi dalam desain skema menghasilkan keunggulan performa yang tidak hanya besar secara magnitude tetapi juga stabil lintas skenario query. Dengan demikian, ketiga model terbukti beroperasi pada kelas performa yang berbeda secara fundamental, bukan sekadar berbeda secara statistik. $\alpha = 0{,}05$; $n = 10$ per kelompok [11], [23], [24].

4.7. Keterbatasan Penelitian

Beberapa keterbatasan penelitian ini perlu diakui untuk memberikan konteks interpretasi hasil yang tepat. Pertama, pengujian dilakukan pada satu sistem manajemen basis data (MySQL 8.0) sehingga hasil tidak dapat digeneralisasi langsung ke DBMS lain seperti PostgreSQL, MariaDB, atau MongoDB. Kedua, seluruh eksperimen dijalankan pada satu perangkat keras kelas konsumen (AMD Ryzen 5 7430U, RAM 8 GB, SSD 256 GB); performa aktual pada server produksi dengan spesifikasi lebih tinggi atau konfigurasi berbeda dapat menghasilkan rasio yang berbeda. Ketiga, pengujian hanya mencakup operasi baca (SELECT) dalam kondisi singl user tanpa beban konkuren. Dampak generated columns terhadap operasi tulis (INSERT, UPDATE, DELETE) dan skenario multi user belum dievaluasi; overhead pemeliharaan indeks pada setiap perubahan data dapat mengurangi keuntungan performa yang diamati pada sisi baca. Keempat, penelitian ini belum mengeksplorasi lingkungan terdistribusi, replikasi MySQL, maupun platform cloud yang dapat memiliki karakteristik I/O dan latensi berbeda secara signifikan. Kelima, profil hardware level seperti utilisasi CPU, konsumsi memori buffer pool, serta metrik disk I/O belum diukur secara eksplisit, sehingga belum dapat disimpulkan apakah keunggulan JSON_GC bersumber dari penghematan I/O, efisiensi cache, atau kombinasi keduanya. Keterbatasan keterbatasan ini membuka ruang bagi penelitian lanjutan yang lebih komprehensif.

5. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa model JSON dengan generated columns dan indeks (JSON_GC) secara konsisten memberikan performa query terbaik dibandingkan model relasional konvensional dan model JSON tanpa optimasi pada seluruh skenario yang diuji, dengan keunggulan bervariasi dari 1,7 kali pada query selektivitas rendah hingga lebih dari 1.200 kali pada query agregasi GROUP BY bergantung pada selektivitas kondisi filter dan jenis operasi yang dilakukan. Perbedaan performa ini bersumber dari kemampuan generated columns untuk mendukung indeks B-Tree reguler yang memungkinkan optimizer MySQL memilih strategi akses ref, range, dan index serta mencapai kondisi covering index yang optimal [7], sementara model JSON_Raw terbukti paling lambat di semua skenario akibat overhead fungsi JSON_EXTRACT dan ketidakmampuannya diindeks secara langsung sehingga selalu memaksa full table scan pada lebih dari satu juta baris. Kontribusi utama penelitian ini terletak pada pembuktian empiris bahwa pendekatan hybrid, yakni penyimpanan data semi-terstruktur dalam kolom JSON yang dikombinasikan dengan generated columns berindeks, merupakan strategi yang layak secara teknis untuk

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

menjembatani kebutuhan fleksibilitas skema dan performa query tinggi dalam satu arsitektur basis data relasional, sehingga memperluas temuan Saputra et al. [7] dan Yusfa et al. [8] yang sebelumnya hanya mengevaluasi pengindeksan pada kolom relasional konvensional. Implikasi praktisnya adalah pengembang sistem informasi sebaiknya menerapkan generated columns pada atribut JSON yang sering digunakan sebagai filter atau pengelompokan disertai indeks B-Tree, sementara penggunaan JSON_Raw tanpa optimasi sebaiknya dihindari untuk skenario query intensif pada dataset berskala jutaan baris [8], [24]. Untuk penelitian lanjutan, evaluasi dampak generated columns terhadap operasi DML (INSERT, UPDATE, DELETE) perlu dilakukan secara terukur mengingat overhead pemeliharaan indeks berpotensi mengurangi keunggulan pada sisi baca [6], perbandingan antara generated columns dan functional index pada MySQL 8.0 perlu dieksplorasi untuk menentukan kondisi optimal penggunaan masing-masing mekanisme, serta replikasi eksperimen pada lingkungan multi-user dan pengujian pada DBMS lain seperti PostgreSQL dengan dukungan native tipe data JSONB dan indeks GIN diperlukan untuk memperluas generalisabilitas temuan ini [17].

6. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas, dosen pembimbing, serta rekan-rekan yang telah memberikan dukungan, bimbingan, dan bantuan data sehingga penelitian ini dapat terselesaikan dengan baik.

DAFTAR PUSTAKA:

[1] M. Lutfi *et al.*, *Konsep Dasar Analisis dan Perancangan Sistem Informasi*. PT Penerbit Qriset Indonesia, 2025.

[2] A. Satria, Z. Zulham, and S. Salamah, "Optimalisasi Transformasi Data Semi-Terstruktur dan Tidak Terstruktur Berbasis XML/JSON dalam Arsitektur Data Warehouse Modern," *Djtechno J. Teknol. Inf.*, vol. 6, no. 2, pp. 820–835, 2025.

[3] B. Suriansyah, L. F. Mz, A. I. Rachman, and G. Pratiwi, "Rekontruksi Arsitektur DataBase untuk Peningkatan Proses Load Data," *J. Media Inform.*, vol. 6, no. 2, pp. 1455–1460, 2025.

[4] A. Amalia and M. A. Rojabi, *MEMBANGUN API MODERN DENGAN JSON*. Afdan Rojabi Publihser, 2026.

[5] A. Turmudi, F. Lanang, M. R. A. Pratama, F. N. Ihsan, D. A. Nugroho, and I. P. Pujiono, "Analisis Perbandingan Struktur Data Dan Kompleksitas Koding Antara MySQL Dan MongoDB Pada Pengembangan Aplikasi Blog," *J. Sains Inform. Terap.*, vol. 5, no. 1, pp. 10–17, 2026.

[6] D. A. E. Saputri, P. D. Lestari, and S. Mukaromah, "Analisis Pengaruh Implementasi Index B-Tree terhadap Kinerja Waktu Database," in *Prosiding Seminar Nasional Teknologi dan Sistem Informasi*, 2023, pp. 475–481.

[7] D. A. Saputra, M. A. Farich, M. N. Anugerah, I. P. Pujiono, and D. A. Nugroho, "Perbandingan Kinerja MongoDB dan MySQL pada Aplikasi dengan Beban Data Besar," *SAINSTECH J. Penelit. DAN Pengkaj. SAINS DAN Teknol.*, vol. 35, no. 4, pp. 71–78, 2025.

[8] M. N. S. Yusfa, A. Musyawwa, and A. B. Pratama, "Analisis Penerapan Indexing dan Query Optimization pada Database MySQL untuk Meningkatkan Performa Aplikasi Absensi Mahasiswa Universitas Pamulang," *J. Inf. Syst. Bus. Technol.*, vol. 1, no. 4, pp. 9–14, 2025.

[9] E. Wihardjo, "DENGAN SOFTWARE MODERN BAR•," *Manaj. Data Dengan Softw. Mod.*, vol. 37, 2025.

[10] D. Van Landuyt, M. Levrau, V. Reniers, and W. Joosen, "An e-commerce benchmark for evaluating performance trade-offs in document stores," in *International Conference on Big Data Analytics and Knowledge Discovery*, Springer, 2024, pp. 284–290.

[11] A. Suciani, D. Ruhiat, and S. D. Rahayu, "Komparasi hasil analisis beda rata-rata menggunakan metode statistik parametrik dan nonparametrik," *JRMST| J. Ris. Mat. Dan Sains Terap.*, vol. 2, no. 2, 2022.

[12] U. Firdaus, A. S. Rini, A. A. Ghifari, A. A. D. Saputra, M. N. Syahwaludin, and D. R. Aliandy, "ANALISIS DAN OPTIMASI PADA DATABASE RELASIONAL MENGGUNAKAN INDEKS DAN NORMALISASI TINGKAT LANJUT: STUDI KASUS SISTEM INFORMASI AKADEMIK," *SIGARUDA J. J. Ilm. Bid. Sos. Ekon. Budaya,*

Jurnal Manajemen Informatika & Sistem Informasi (MISI)
Volume 9, Nomor 2, Juni 2026.
ISSN 2614-1701 (media cetak)
ISSN 2614-3739 (media online)
DOI : 10.36595/misi.v5i2
http://e-journal.stmiklombok.ac.id/index.php/misi

Teknol. dan Pendidik., vol. 1, no. 2, pp. 566–571, 2025.

[13] A. K. Wijaya, "Analisis Struktur Basis Data pada Aplikasi E-Commerce Tokopedia: Studi Kualitatif terhadap Desain dan Optimalisasi Skema Relasional," *SAINSTECH J. Penelit. DAN Pengkaj. SAINS DAN Teknol.*, vol. 35, no. 2, pp. 50–57, 2025.

[14] G. Yuan, J. Lu, Z. Yan, and S. Wu, "A survey on mapping semi-structured data and graph data to relational data," *ACM Comput. Surv.*, vol. 55, no. 10, pp. 1–38, 2023.

[15] A. Fadli, M. I. Zulfa, A. W. W. Nugraha, A. Taryana, and M. S. Aliim, "Analisis perbandingan unjuk kerja database sql dan database nosql untuk mendukung era big data," *J. Nas. Tek. Elektro*, pp. 154–158, 2020.

[16] J. I. Thirupattur, A. X. T. Tan, S.-L. Lim, L. H.-E. Wong, M. J. Yong, and S. V. Easwaramoorthy, "Comparative Performance Analysis of SQL and NoSQL Databases for Optimizing Retail and E-Commerce Operations," in *2024 9th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, 2024, pp. 1063–1070.

[17] M. Salahuddin, S. Majeed, S. Hira, and G. Mumtaz, "A systematic literature review on performance evaluation of sql and nosql database architectures," *J. Comput. Biomed. Informatics*, vol. 7, no. 02, 2024.

[18] H. Haerullah, "PERBANDINGAN EFISIENSI QUERY DATABASE RELASIONAL DAN NOSQL PADA APLIKASI E-COMMERCE: STUDI EKSPERIMEN," *J. Rekayasa Teknol. Inf.*, vol. 9, no. 4, pp. 386–395, 2025.

[19] L. Zhang, K. Pang, J. Xu, and B. Niu, "JSON-based control model for SQL and NoSQL data conversion in hybrid cloud database," *J. Cloud Comput.*, vol. 11, no. 1, p. 23, 2022.

[20] M. N. Ilham, A. F. Setiawan, I. Prawira, D. Purnomo, D. A. Nugroho, and I. P. Pujiono, "EVALUASI PERFORMA MYSQL DAN MONGODB PADA QUERY PATTERN E-COMMERCE SKALA KECIL BERBASIS LITERATUR," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 10, no. 1, pp. 15–21, 2026.

[21] T. R. Hadyan and S. Wahyu, "OPTIMALISASI KINERJA QUERY DAN BACKEND SISTEM PELAPORAN PENUNJANG MEDIS PADA SISTEM INFORMASI MANAJEMEN RUMAH SAKIT," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 10, no. 2, pp. 3210–3220, 2026.

[22] H. Utomo, "Perbandingan tabel mortalita indonesia dan tabel mortalita cso menggunakan uji mann-whitney dan uji kruskal-wallis," *Syntax Lit. J. Ilm. Indones.*, vol. 6, no. 3, p. 1210, 2021.

[23] P. Kosamkar, G. Sharma, L. Upadhyaya, B. Shah, and S. Patel, "Query Optimization: Techniques and Strategies for MySQL Performance Improvement," in *International Conference on ICT for Sustainable Development*, Springer, 2025, pp. 134–142.

[24] T. F. Llano-Rios, "Using dynamic schemas for query optimization over JSON data.," 2024.