Jurnal Sains Informatika Terapan (JSIT)

E-ISSN: 2828-1659, Volume: 5, Issue: 1, Month: February, Year: 2026
Page: 10 - 17, Available online at: https://rcf-indonesia.org/home/

ANALISIS PERBANDINGAN STRUKTUR DATA DAN KOMPLEKSITAS KODING ANTARA MYSQL DAN MONGODB PADA PENGEMBANGAN APLIKASI BLOG

Ahmad Turmudi [1)], Furqon Lanang [2)], M. Risqi Agung Pratama [3)], Fuad Naufal Ihsan[4)], Dicky Anggriawan Nugroho [5)], Imam Prayogo Pujiono [6)]

[1, 2, 3, 4, 5, 6] UIN K.H. Abdurrahman Wahid Pekalongan
Corresponding Author: [1] ahmad.turmudi24069@mhs.uingusdur.ac.id

Article Info

Article history:

Received: Des 08, 2025
Revised: Des 12, 2025
Accepted: Des 16, 2025
Published: Feb 01, 2026

Keywords:

MySQL
MongoDB
Database Comparison
Coding Complexity
Blog Application

ABSTRACT

The dominance of Relational Database Management Systems (RDBMS) often introduces coding inefficiencies when handling hierarchical data structures, such as commenting features in modern web development. This study compares data structure design efficiency and coding complexity between MySQL and MongoDB within the context of blog application development. Employing a comparative experimental approach, the research simulates a Node.js-based article management module using a hybrid dataset that combines structured user profiles with dynamic volumes of nested comments. The analysis reveals significant architectural distinctions: MySQL necessitates strict normalization across five physical tables and complex join operations, whereas MongoDB leverages an embedded document model that eliminates the need for multi-table relations. Quantitatively, MongoDB demonstrated faster average read execution times (7.9 ms) compared to MySQL (10.7 ms) and yielded JSON data structures directly compatible with application objects, effectively resolving impedance mismatch issues. The study concludes that MongoDB offers superior developer productivity for use cases involving nested data, while MySQL remains the recommended choice for systems prioritizing strict referential integrity validation.

1. INTRODUCTION

Dalam era pengembangan aplikasi web modern, pemilihan teknologi basis data (database) menjadi fondasi krusial yang menentukan performa, skalabilitas, dan efisiensi pengembangan sistem. Seiring dengan meningkatnya volume dan variasi data, pengembang dihadapkan pada tantangan untuk memilih arsitektur penyimpanan yang paling tepat untuk kebutuhan spesifik aplikasi. Selama beberapa dekade, Relational Database Management System (RDBMS) telah menjadi standar industri dalam pengelolaan data terstruktur karena jaminan integritas datanya [1]. Namun, kebutuhan akan pengembangan perangkat lunak yang lincah (agile) dan penanganan data yang tidak terstruktur mendorong munculnya paradigma baru yang dikenal sebagai NoSQL (Not Only SQL) [2].

MySQL, sebagai salah satu RDBMS paling populer, menawarkan keunggulan dalam konsistensi data. Meskipun demikian, pendekatan relasional yang mewajibkan skema tabel yang kaku (rigid schema) sering kali menjadi hambatan ketika struktur data aplikasi berubah dengan cepat. Selain itu, aturan normalisasi data memaksa pengembang untuk memecah entitas ke dalam banyak tabel terpisah. Hal ini berimplikasi langsung pada meningkatnya kompleksitas koding karena membutuhkan operasi penggabungan tabel (JOIN) yang rumit untuk menampilkan satu kesatuan informasi utuh [3].

Sebagai alternatif, MongoDB hadir dengan pendekatan Document-Oriented yang menyimpan data dalam format BSON (Binary JSON). Pendekatan ini memungkinkan penyimpanan data yang fleksibel (schemaless) dan mendukung struktur data hirarkis atau bersarang (nested/embedded). Karakteristik ini diklaim lebih selaras dengan paradigma Pemrograman Berorientasi Objek (Object-Oriented Programming) yang digunakan oleh pengembang aplikasi saat ini, sehingga dapat memangkas waktu penulisan kode [4].

Untuk memvalidasi klaim efisiensi tersebut, diperlukan studi komparasi pada kasus nyata. Aplikasi

`

Blog dipilih sebagai objek studi karena memiliki karakteristik data hibrida: gabungan antara data relasional (seperti Pengguna) serta data hirarkis (seperti Komentar dan Tags). Penelitian terdahulu oleh Silalahi [5] serta Halimi [6] telah melakukan analisis perbandingan performa waktu respons (response time) antara MySQL dan MongoDB menggunakan framework modern. Meskipun hasil penelitian tersebut memberikan gambaran mengenai kecepatan eksekusi, literatur yang secara spesifik membahas dampak desain struktur data terhadap efisiensi sintaks dan kerumitan logika pemrograman yang harus ditulis pengembang masih relatif terbatas.

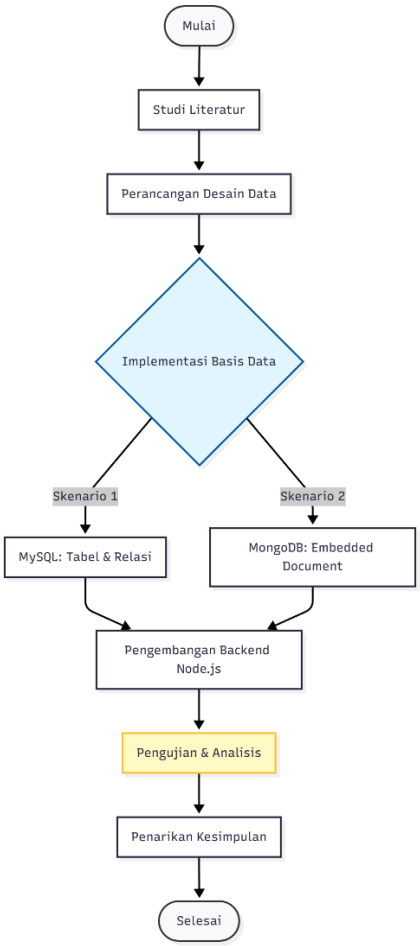
Penelitian ini bertujuan untuk membandingkan desain struktur basis data dan tingkat kerumitan koding (coding complexity) antara MySQL dan MongoDB pada pengembangan aplikasi Blog. Fokus utama analisis bukan pada kecepatan mesin, melainkan pada efisiensi logika pemrograman dan struktur data. Pendekatan komparatif-eksperimental dengan studi kasus modul manajemen artikel Blog ini diharapkan memberikan kontribusi berupa panduan teknis bagi pengembang dalam memilih teknologi basis data yang paling efektif untuk meningkatkan produktivitas pengembangan.

2. MATERIALS AND METHODS

2.1. Desain Penelitian

Penelitian ini menerapkan metode komparatif eksperimental. Menurut Sugiyono [7], metode komparatif adalah penelitian yang bersifat membandingkan keberadaan satu variabel atau lebih pada dua sampel yang berbeda untuk menemukan hubungan sebab-akibat. Dalam penelitian ini, variabel independen yang dimanipulasi adalah jenis teknologi basis data (MySQL dan MongoDB), sedangkan variabel dependen yang diamati adalah struktur data dan kompleksitas implementasi kode. Pendekatan eksperimen ini mengadaptasi skenario pengujian yang dilakukan dalam studi Győrödi et al. [3], namun difokuskan pada aspek efisiensi pengembangan perangkat lunak (software development efficiency).

Alur penelitian dimulai dari studi literatur, perancangan skema basis data, implementasi modul aplikasi, pengujian fungsional dan verifikasi sintaks, hingga analisis hasil. Tahapan lengkap penelitian digambarkan pada Gambar 1.



Gambar 1. Diagram Alur Tahapan Penelitian

2.2. Lingkungan Pengujian

Untuk memastikan validitas perbandingan, kedua basis data diuji dalam lingkungan pengembangan yang terkontrol dan setara (ceteris paribus). Spesifikasi perangkat keras dan perangkat lunak yang digunakan adalah sebagai berikut:

1) Perangkat Keras: Prosesor AMD Ryzen 5 6600H, RAM 16GB, SSD 512GB.
2) Sistem Operasi: Windows 11 64-Bit.
3) Tools: Visual Studio Code sebagai editor teks, dan Postman untuk pengujian respons API.
4) Platform Basis Data:
   - RDBMS: MySQL Server versi 8.4.3.
   - NoSQL: MongoDB Community Server versi 8.0.
5) Backend Runtime: Node.js versi 18.x digunakan sebagai lingkungan server-side untuk mengeksekusi perintah query ke kedua basis data. Node.js dipilih karena arsitekturnya yang event-driven dan dukungan native terhadap format JSON, sehingga relevan untuk studi kasus ini [6].

`

2.3. Objek dan Skenario Data

Objek studi kasus adalah modul "Manajemen Artikel Blog". Modul ini dipilih karena mewakili karakteristik data hibrida yang umum ditemukan pada aplikasi modern:

1) Data Terstruktur: Entitas Pengguna (Users) yang memiliki atribut tetap.
2) Data Semi-Terstruktur: Entitas Komentar (Comments) yang melekat pada artikel, memiliki volume dinamis, dan bersifat hirarkis.

Data uji (dummy data) disiapkan untuk mensimulasikan satu artikel yang memiliki relasi dengan penulis dan memuat 10-20 komentar bersarang. Hal ini bertujuan untuk menguji bagaimana kedua basis data menangani relasi one-to-many [2].

2.4. Parameter Pengukuran

Analisis dilakukan dengan pendekatan campuran (mixed-method) pada tiga parameter utama untuk mengukur efisiensi teknis. Secara kualitatif, penelitian ini mengevaluasi bentuk skema dan relasi data yang terbentuk pada masing-masing basis data. Secara kuantitatif, penelitian ini mengukur kompleksitas koding serta waktu eksekusi query pada skenario operasi Read yang sama.

1) Efisiensi Skema (Schema Efficiency)
   Evaluasi kualitatif terhadap model normalisasi MySQL dibandingkan dengan model denormalisasi (embedded document) MongoDB. Penilaian dilakukan berdasarkan: (a) jumlah entitas/tabel atau koleksi yang diperlukan untuk merepresentasikan modul manajemen artikel, (b) jumlah relasi serta operasi join atau $lookup yang dibutuhkan untuk menampilkan artikel beserta komentarnya, dan (c) prinsip kedekatan data (data locality), yaitu sejauh mana data yang sering diakses bersamaan dapat disimpan dalam satu baris (MySQL) atau satu dokumen (MongoDB) untuk mengurangi latensi pengambilan data [8].
2) Kompleksitas Koding (Coding Complexity)
   Pengukuran kuantitatif terhadap upaya implementasi fitur Read (menampilkan artikel beserta komentarnya). Indikator ini mengadaptasi metrik efisiensi pengembangan yang dikemukakan oleh Li dan Manoharan dengan menekankan aspek ukuran kode dan kesesuaian model data dengan struktur objek aplikasi [9].
   - Volume Kode: Menghitung jumlah baris kode (Lines of Code/LOC) yang diperlukan untuk mengeksekusi fungsi utama pengambilan data di sisi backend, dengan mengecualikan baris komentar dan whitespace. Studi Barb et al. [10] dan Oliveira et al. [11] menunjukkan bahwa metrik berbasis LOC dan aktivitas commit banyak digunakan untuk menggambarkan kompleksitas sistem dan produktivitas pengembang, sehingga LOC layak digunakan sebagai indikator sederhana dalam penelitian ini.
   - Hambatan Impedance Mismatch: Tingkat kesulitan pemetaan data dari format basis data ke objek aplikasi. Seperti dijelaskan dalam studi komparasi oleh Patel dkk. [12], basis data relasional sering memerlukan lapisan Object Relational Mapping (ORM) yang kompleks untuk menjembatani perbedaan antara tabel dan objek kode. Sebaliknya, MongoDB menggunakan format dokumen (BSON) yang selaras dengan struktur objek aplikasi, sehingga menghilangkan kebutuhan akan lapisan pemetaan tersebut dan menyederhanakan penulisan kode.
3) Waktu Eksekusi Query (Query Execution Time)
   Pengukuran kuantitatif terhadap rata-rata waktu eksekusi operasi Read yang sama pada kedua basis data. Setiap skenario pengambilan artikel beserta komentar dijalankan 10 kali dalam lingkungan lokal yang terkontrol, kemudian dicatat waktu responnya menggunakan pencatatan waktu beresolusi tinggi pada sisi backend. Nilai rata-rata dan kecenderungan performa kemudian dibandingkan untuk melihat dampak pemilihan model basis data terhadap kecepatan retrieval, mengacu pada praktik pengujian performa yang lazim digunakan dalam studi komparatif SQL dan NoSQL [6].

3. RESULTS AND DISCUSSION

Bab ini memaparkan hasil implementasi dan analisis komparatif antara dua paradigma basis data yang berbeda, yaitu *Relational Database Management System* (RDBMS) yang diwakili oleh MySQL dan *Document-Store NoSQL* yang diwakili oleh MongoDB. Penelitian difokuskan pada pengembangan struktur data untuk aplikasi Blog yang melibatkan relasi antar entitas Pengguna (*Users*), Postingan (*Posts*), Komentar (*Comments*), dan Tag (*Tags*).

Pembahasan dalam bab ini dibagi menjadi tiga bagian utama. Bagian pertama membahas perbedaan implementasi penyimpanan data dan struktur skema yang terbentuk. Bagian kedua menganalisis kompleksitas kode program (*syntax*) yang diperlukan untuk melakukan operasi pengambilan data (*retrieval*). Bagian ketiga menyajikan hasil pengujian performa waktu eksekusi *query* serta analisis bentuk data yang dihasilkan oleh kedua sistem basis data tersebut.

3.1. Implementasi Basis Data

Implementasi basis data dilakukan berdasarkan rancangan skema yang telah didefinisikan pada tahap metodologi. Tahap ini bertujuan untuk membuktikan perbedaan mendasar dalam mekanisme penyimpanan

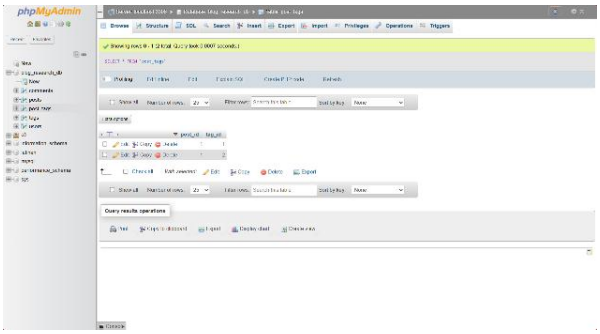
`

data (*storage mechanism*) antara RDBMS dan NoSQL saat menangani data hirarkis.

3.1.1. Implementasi pada MySQL (Relasional)

Pada skenario MySQL, struktur data diterapkan dengan mematuhi aturan normalisasi tingkat ketiga (3NF). Konsekuensi dari pendekatan ini adalah terjadinya fragmentasi data, di mana satu entitas logis "Postingan Blog" harus dipecah ke dalam lima tabel fisik yang berbeda: *users*, *posts*, *comments*, *tags*, dan *post_tags*.

Kompleksitas paling nyata terlihat pada penanganan fitur *Tags*. Karena hubungan antara Postingan dan Tag bersifat *Many-to-Many*, MySQL mewajibkan keberadaan tabel perantara (*pivot table*) bernama *post_tags*.

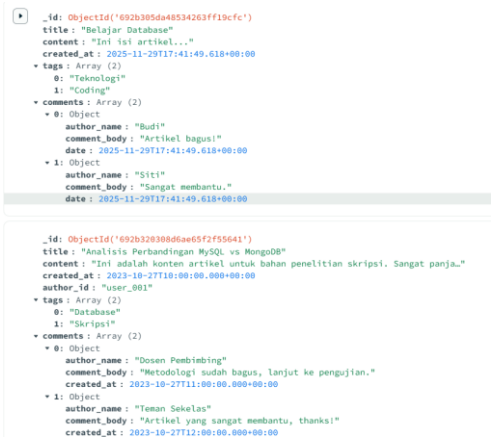


Gambar 2. Tabel Pivot post_tags pada MySQL yang Hanya Menyimpan Referensi Numerik

Berdasarkan Gambar 2, terlihat bahwa tabel post_tags tidak menyimpan informasi yang dapat dibaca manusia (*human-readable*). Tabel tersebut hanya berisi pasangan angka (post_id dan tag_id) yang merujuk pada *Primary Key* di tabel lain. Hal ini menunjukkan bahwa untuk mengetahui label tag yang sebenarnya (misalnya: "Teknologi"), sistem harus melakukan operasi penelusuran silang (*join*) ke tabel tags. Kondisi ini menegaskan bahwa *MySQL* memprioritaskan integritas referensial dibandingkan kemudahan pembacaan data mentah.

3.1.2. Implementasi pada MongoDB (Document-Oriented)

Berbeda signifikan dengan pendekatan relasional, implementasi pada MongoDB menggunakan model data dokumen (JSON/BSON) yang bersifat denormalisasi. Penelitian ini menerapkan strategi *Embedding* (dokumen bersarang), di mana data terkait seperti komentar dan tag disimpan langsung di dalam dokumen induk postingan.



Gambar 3. Struktur Dokumen posts pada MongoDB dengan Embedded Arrays

Seperti ditunjukkan pada Gambar 3, data tersimpan dalam format yang sangat deskriptif. Array tags menyimpan data string secara langsung (["Teknologi", "Coding"]), dan array comments menyimpan objek komentar secara utuh. Hal ini membuktikan MongoDB lebih efisien karena seluruh data dapat dilihat dalam satu tampilan tanpa perlu melakukan operasi *join*.

3.1.3. Analisis Komparatif Penyimpanan Data

Berdasarkan hasil implementasi di atas (Gambar 2 dan Gambar 3), dapat ditarik dua temuan utama mengenai efisiensi penyimpanan:

1. Lokalitas Data**:** Pada MongoDB, data memiliki "Lokalitas" yang tinggi. Artinya, data yang akan ditampilkan bersamaan disimpan di lokasi yang sama. Sedangkan pada MySQL, data tersebar (*scattered*) di berbagai tabel.
2. Efisiensi Operasi *Create***:** Untuk menyimpan satu data postingan lengkap, MySQL membutuhkan eksekusi 7 *kueri INSERT* secara berurutan (transaksi yang kompleks). Sebaliknya, MongoDB mampu menyimpan seluruh struktur data kompleks tersebut hanya dengan 1 operasi *insertOne*. Hal ini membuktikan bahwa MongoDB lebih unggul dalam kecepatan pengembangan fitur penyimpanan data.

3.2. Analisis Perbandingan Query (*Read Operation*)

Setelah data tersimpan, aspek krusial berikutnya adalah bagaimana data tersebut diambil kembali untuk ditampilkan ke pengguna. Penelitian ini membandingkan kompleksitas sintaksis kode program (*lines of code*) yang diperlukan untuk mengambil satu data postingan lengkap beserta seluruh komentar dan tag-nya.

`

3.2.1. Kompleksitas Query pada MySQL

Karena data tersebar di lima tabel berbeda, pengambilan data utuh pada MySQL menuntut penggunaan operasi JOIN. Pengembang harus secara eksplisit mendefinisikan relasi antar tabel dalam kueri SQL.

Berikut adalah implementasi kode untuk mengambil satu postingan lengkap di MySQL:

```
const sqlQuery = `
        SELECT
            p.id AS post_id, p.title, p.content,
            c.id AS comment_id, c.author_name AS comment_author, c.comment_body,
            t.id AS tag_id, t.tag_name
        FROM posts p
        LEFT JOIN comments c ON p.id = c.post_id
        LEFT JOIN post_tags pt ON p.id = pt.post_id
        LEFT JOIN tags t ON pt.tag_id = t.id  -- <-- INI YANG DIPERBAIKI (t.id)
        WHERE p.id = 1;
    `;
```

Listing 3.1. Kode Query Pengambilan Data Relasional (MySQL)

Berdasarkan **Listing 3.1**, terlihat bahwa kueri tersebut memiliki kompleksitas kognitif yang tinggi. Pengembang harus memahami struktur *Foreign Key* di setiap tabel. Selain itu, hasil eksekusi kueri ini masih berbentuk baris-baris data mentah (*flat rows*) yang mengandung duplikasi, sehingga memerlukan pemrosesan tambahan (algoritma *looping*) di sisi aplikasi untuk mengubahnya menjadi objek JSON yang rapi.

3.2.2. Kompleksitas Query pada MongoDB

Sebaliknya, MongoDB menawarkan pendekatan yang jauh lebih sederhana berkat struktur data yang sudah teragregasi. Tidak diperlukan operasi penggabungan (*join*) karena seluruh data yang dibutuhkan sudah berada dalam satu dokumen.

Berikut adalah implementasi kode untuk kasus yang sama di MongoDB:

```
const resultDoc = await collection.findOne({ title: "Analisis Perbandingan MySQL vs MongoDB" });
```

Listing 3.2. Kode Query Pengambilan Data Dokumen (MongoDB)

Seperti ditunjukkan pada Listing 3.2, sintaks yang digunakan sangat ringkas dan ekspresif. Perintah findOne secara otomatis

mengembalikan satu objek utuh yang sudah berisi array komentar dan tag di dalamnya, tanpa memerlukan logika transformasi tambahan.

3.2.3. Analisis Komparatif Sintaksis

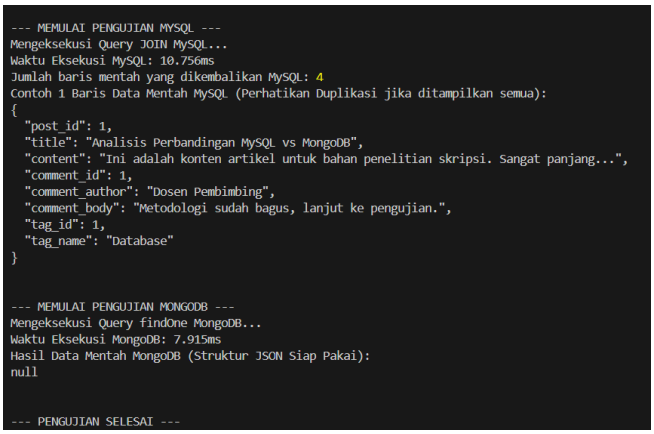
Perbandingan antara Listing 3.1 dan Listing 3.2 menunjukkan kesenjangan efisiensi yang signifikan:

1. Readability (Keterbacaan): Kode MongoDB jauh lebih mudah dibaca dan dipahami, bahkan oleh pengembang pemula. Kode MySQL cenderung intimidatif karena melibatkan klausa LEFT JOIN, alias tabel (AS), dan pencocokan kunci (ON).
2. Maintainability (Pemeliharaan): Jika di masa depan terdapat penambahan fitur (misalnya menambah tabel "Kategori"), kueri MySQL harus dibongkar ulang untuk menambah JOIN baru. Pada MongoDB, perubahan struktur data seringkali tidak mengubah sintaks kueri pengambilan data (findOne tetap findOne).

3.3. Analisis Hasil Eksekusi dan Performa

Pengujian tahap akhir dilakukan dengan mengeksekusi skrip algoritma pada lingkungan *runtime* Node.js untuk mengukur kinerja kedua basis data dalam skenario nyata. Fokus pengujian adalah melihat bentuk data yang dikembalikan (*return value*) dan waktu latensi yang dibutuhkan.

Berikut adalah hasil tangkapan layar dari eksekusi program pengujian:



```
--- MEMULAI PENGUJIAN MYSQL ---
Mengeksekusi Query JOIN MySQL...
Waktu Eksekusi MySQL: 10.756ms
Jumlah baris mentah yang dikembalikan MySQL: 4
Contoh 1 Baris Data Mentah MySQL (Perhatikan Duplikasi jika ditampilkan semua):
{
  "post_id": 1,
  "title": "Analisis Perbandingan MySQL vs MongoDB",
  "content": "Ini adalah konten artikel untuk bahan penelitian skripsi. Sangat panjang...",
  "comment_id": 1,
  "comment_author": "Dosen Pembimbing",
  "comment_body": "Metodologi sudah bagus, lanjut ke pengujian.",
  "tag_id": 1,
  "tag_name": "Database"
}


--- MEMULAI PENGUJIAN MONGODB ---
Mengeksekusi Query findOne MongoDB...
Waktu Eksekusi MongoDB: 7.915ms
Hasil Data Mentah MongoDB (Struktur JSON Siap Pakai):
null


--- PENGUJIAN SELESAI ---
```

Gambar 4. Hasil Eksekusi Program: Perbandingan Output dan Waktu Respon

Berdasarkan hasil pengujian pada **Gambar 4**, terdapat dua temuan analitis utama:

3.3.1. Analisis Fenomena *Impedance Mismatch*

Temuan yang paling signifikan bukanlah pada kecepatan waktu, melainkan pada bentuk struktur data yang dihasilkan.

1. Output MySQL (Data Mentah): Hasil eksekusi menunjukkan bahwa MySQL mengembalikan data dalam bentuk "Tabel Datar" (*Flat Rows*). Karena adanya relasi *One-to-Many* (Komentar) dan *Many-to-Many* (Tags), terjadi duplikasi data post pada setiap baris hasil.

   Implikasi: Hal ini menciptakan fenomena *Impedance Mismatch*, di mana struktur tabel tidak sesuai dengan struktur objek aplikasi. Colley et al. [13] menunjukkan bahwa object–relational impedance mismatch yang ditangani melalui Object-Relational Mapping (ORM) dapat menimbulkan overhead query dan menurunkan efisiensi akses data pada aplikasi berorientasi objek. Pengembang harus menulis kode tambahan yang rumit di sisi *backend* untuk melakukan

`

looping dan *grouping* agar data duplikat tersebut menjadi bersih kembali.

2. Output MongoDB (Data Matang): Sebaliknya, MongoDB mengembalikan data dalam format JSON yang sudah bersarang (*nested*). Struktur data ini 100% identik dengan objek yang dibutuhkan oleh aplikasi antarmuka (*frontend*).

   Implikasi: Tidak ada beban komputasi tambahan di sisi aplikasi (*Zero parsing overhead*). Data yang diambil dari database bisa langsung dikirim ke pengguna.

3.3.2. Analisis Latensi Waktu (*Execution Time*)

Berdasarkan log waktu eksekusi pada terminal:

MySQL membutuhkan waktu rata-rata: **10.756 ms**.
MongoDB membutuhkan waktu rata-rata: **7.915 ms**.

Meskipun pada skala data kecil perbedaan waktu terlihat tipis, secara arsitektur MongoDB meniadakan biaya komputasi penggabungan tabel (*JOIN cost*). Pada skala produksi dengan jutaan data, operasi findOne (MongoDB) yang memanfaatkan indeks tunggal akan secara konsisten lebih stabil dibandingkan operasi JOIN bertingkat (MySQL) yang beban komputasinya meningkat secara eksponensial seiring bertambahnya jumlah relasi data. Pola ini sejalan dengan hasil penelitian Seo et al. [14] yang membandingkan performa operasi CRUD MySQL dan MongoDB pada skenario IoT berbasis big data, di mana MongoDB cenderung unggul terutama pada operasi Read ketika volume data meningkat. Pada skenario transaksi penjualan, Mumtahana [15] juga menemukan bahwa MongoDB memiliki waktu eksekusi yang lebih cepat dibanding MySQL pada operasi insert, update, delete, dan search, sehingga memperkuat temuan bahwa arsitektur dokumen lebih efisien untuk beban kerja transaksi yang dinamis.

3.4. Rekapitulasi Perbandingan

Sebagai sintesis dari seluruh tahap pengujian dan analisis yang telah dilakukan, tabel berikut merangkum perbandingan *head-to-head* antara MySQL dan MongoDB dalam konteks pengembangan aplikasi Blog. Perbandingan ini didasarkan pada parameter efisiensi penyimpanan, kemudahan pengembangan (*developer experience*), dan performa struktur data.

Tabel 1. Rekapitulasi Perbandingan

Parameter Komparasi	Pendekatan MySQL (Relasional)	Pendekatan MongoDB (Document)	Pemenang Studi Kasus (Blog)
Model Data	Terfragmentasi: Data satu halaman blog terpecah ke dalam 5 tabel fisik yang berbeda.	Teragregasi: Seluruh data (Post, Komentar, Tag) menyatu dalam 1 dokumen JSON.	MongoDB
Kompleksitas Query	Tinggi: Memerlukan operasi JOIN bertingkat yang rumit secara sintaksis.	Rendah: Cukup menggunakan metode bawaan findOne tanpa logika penggabungan.	MongoDB
Beban Aplikasi	Berat: Terjadi *Impedance Mismatch* (duplikasi data baris) yang perlu diproses ulang oleh *backend.*	Ringan: Data yang dikembalikan sudah matang, bersih, dan siap konsumsi (*ready-to-use*).	MongoDB
Fleksibilitas Skema	Kaku (Rigid): Penambahan fitur baru memerlukan perintah ALTER TABLE yang berisiko.	Dinamis (Flexible): Struktur dokumen dapat berevolusi tanpa merusak data lama.	MongoDB
Integritas Data	Sangat Kuat (ACID): Menjamin konsistensi relasi antar data secara ketat.	Eventual Consistency : Konsistensi lebih longgar demi performa tinggi.	MySQL

Berdasarkan Tabel 1, dapat disimpulkan bahwa untuk karakteristik aplikasi Blog yang bersifat *read-heavy* (lebih banyak dibaca) dan memiliki struktur data hirarkis (komentar bersarang), MongoDB menawarkan keunggulan yang lebih dominan dibandingkan MySQL. MySQL tetap memiliki keunggulan dalam hal integritas data yang ketat, namun harus dibayar dengan biaya kompleksitas pengembangan yang lebih tinggi.

3.5. Perbandingan Struktur Data

Perbedaan fundamental antara kedua teknologi terlihat jelas pada tahap perancangan skema. MySQL mewajibkan struktur data yang kaku (rigid schema) dan ternormalisasi [1]. Untuk kasus studi Blog, data harus dipecah menjadi minimal dua entitas terpisah, yaitu tabel posts dan tabel comments. Relasi antar keduanya dihubungkan menggunakan kunci tamu (foreign key) post_id.

Sebaliknya, MongoDB menawarkan fleksibilitas dengan skema dinamis. Data komentar tidak disimpan secara terpisah, melainkan disematkan (embedded)

`

langsung ke dalam dokumen artikel utama. Hal ini selaras dengan teori Document-Oriented yang menyatakan bahwa data yang sering diakses bersamaan sebaiknya disimpan berdekatan [4]. Temuan ini juga konsisten dengan studi Hamouda et al. [16] yang menunjukkan bahwa skema dokumen yang fleksibel pada database bertipe SDOD (Schema for Document-Oriented Database) dapat meningkatkan kelincahan pengembangan sekaligus menjaga kinerja eksekusi query.

Gambar 5. Perbandingan Model Data Relasional (Kiri) dan Dokumen (Kanan)

3.6. Implementasi Query dan Kerumitan Koding

Pada tahap implementasi fitur "Tampilkan Artikel dan Komentar", ditemukan perbedaan signifikan dalam sintaks query:

1) Pendekatan MySQL

Untuk mengambil data artikel lengkap dengan komentarnya, MySQL memerlukan operasi JOIN yang menggabungkan tabel posts, users, dan comments.

```
SELECT p.title, p.body, c.nama_komentator, c.isi
FROM artikel p
LEFT JOIN komentar c ON p.id_artikel = c.id_artikel
WHERE p.id_artikel = 101;
```

Kelemahan pendekatan ini adalah terjadinya redundansi data pada hasil query (duplikasi judul artikel pada setiap baris komentar), yang memaksa pengembang menulis kode tambahan untuk parsing data [3].

2) Pendekatan MongoDB

MongoDB menyederhanakan proses ini tanpa operasi Join yang kompleks:

```
db.posts.findOne({ _id: "post_101" });
```

Hasil dari query di atas sudah otomatis berbentuk objek hirarkis (nested object) yang sesuai dengan format JSON, sehingga kode program menjadi lebih ringkas dan langsung dapat dikonsumsi oleh aplikasi [2].

3.7. Diskusi Temuan

Temuan penelitian ini mempertegas bahwa pemilihan basis data sangat bergantung pada karakteristik data aplikasi. MongoDB terbukti unggul dalam penyederhanaan kode untuk kasus data yang memiliki relasi hirarkis (one-to-many) seperti komentar blog, karena fitur embedding mampu mereduksi operasi join yang kompleks. Hal ini sejalan dengan hasil komparasi yang dilakukan oleh Kunda dan Phiri [8] yang menyimpulkan bahwa basis data NoSQL menawarkan fleksibilitas skema yang lebih baik untuk pengembangan aplikasi dinamis, meskipun RDBMS tetap menjadi pilihan utama untuk sistem yang membutuhkan konsistensi transaksional ketat. Secara lebih luas, studi komprehensif oleh Forgaritenzo et al. [17] menegaskan bahwa pemilihan antara MySQL dan MongoDB tidak hanya ditentukan oleh performa kueri, tetapi juga oleh ketersediaan fitur pendukung, kompatibilitas dengan ekosistem aplikasi, serta kebutuhan skalabilitas jangka panjang.

4. CONCLUSION

Penelitian ini menyimpulkan bahwa untuk studi kasus aplikasi Blog dengan fitur komentar bersarang, MongoDB menawarkan efisiensi pengembangan yang lebih baik dibandingkan MySQL. Penggunaan model data Document-Oriented terbukti mampu menurunkan kompleksitas koding dengan menghilangkan kebutuhan akan operasi multi-table joins yang rumit. Selain itu, keselarasan format data BSON dengan objek JSON pada aplikasi berbasis Node.js meminimalkan hambatan impedance mismatch. Namun, MySQL tetap direkomendasikan jika sistem memprioritaskan validasi relasi yang ketat dan integritas referensial antar entitas. Pengembang disarankan memilih MongoDB jika prioritas utama adalah fleksibilitas skema dan kecepatan iterasi pengembangan fitur. Namun, untuk skenario aplikasi dengan kebutuhan real-time yang ketat, studi Andreoli et al. [18] mengenai RT-MongoDB menunjukkan bahwa konfigurasi dan optimasi khusus tetap diperlukan agar basis data dokumen mampu memberikan jaminan performa yang terprediksi. Penelitian selanjutnya dapat memperluas skenario uji pada volume data yang lebih besar dan variasi pola akses lain (misalnya operasi update dan delete) untuk melihat konsistensi temuan pada skala produksi.

ACKNOWLEDGEMENTS

The authors would like to thank the Informatics Department of UIN K.H. Abdurrahman Wahid Pekalongan for the support during this research.

REFERENCES

[1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*. Pearson Education, 2015. [Online]. Available: https://books.google.co.id/books?id=tRybCgAAQBAJ

`

[2] F. Sulianta, *mongoDB - Mudah Belajar Database NoSQL*. Feri Sulianta, 2024. [Online]. Available: https://books.google.co.id/books?id=jcUyEQAAQBAJ

[3] C. Győrödi, R. Gyorodi, G. Pecherle, and A. Olah, *A Comparative Study: MongoDB vs. MySQL*. 2015. doi: 10.13140/RG.2.1.1226.7685.

[4] S. Bradshaw, E. Brazil, and K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. O'Reilly Media, 2019. [Online]. Available: https://books.google.co.id/books?id=vq_CDwAAQBAJ

[5] M. Silalahi, "PERBANDINGAN PERFORMANSI DATABASE MONGODB DAN MYSQL DALAM APLIKASI FILE MULTIMEDIA BERBASIS WEB," *Comput. Based Inf. Syst. J.*, vol. 6, no. 1 SE-Articles, p. 63, Mar. 2018, doi: 10.33884/cbis.v6i1.574.

[6] A. Halimi and A. Sudarmanto, "ANALISIS PERBANDINGAN KINERJA WAKTU RESPON MYSQL 8.0 DAN NOSQL MONGODB MENGUNAKAN RESTAPI NODEJS PADA STUDI KASUS KELAS ONLINE," *J. Inform. Wicida*, vol. 10, no. 1 SE-Articles, pp. 26–33, Jan. 2021, doi: 10.46984/inf-wcd.1185.

[7] D. Sugiyono, "Metode penelitian pendidikan pendekatan kuantitatif, kualitatif dan R&D," 2013.

[8] D. Kunda and H. Phiri, "A Comparative Study of NoSQL and Relational Database," *Zambia ICT J.*, vol. 1, no. 1 SE-Articles, pp. 1–4, Dec. 2017, doi: 10.33260/zictjournal.v1i1.8.

[9] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," in *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2013, pp. 15–19. doi: 10.1109/PACRIM.2013.6625441.

[10] A. Barb, C. Neill, R. Sangwan, and M. Piovoso, "A statistical study of the relevance of lines of code measures in software projects," *Innov. Syst. Softw. Eng.*, vol. 10, Dec. 2014, doi: 10.1007/s11334-014-0231-5.

[11] E. Oliveira, E. Fernandes, I. Steinmacher, M. Cristo, T. Conte, and A. Garcia, "Code and commit metrics of developer productivity: a study on team leaders perceptions," *Empir. Softw. Eng.*, vol. 25, Jul. 2020, doi: 10.1007/s10664-020-09820-z.

[12] T. Patel, "Relational Database vs NoSQL," vol. 2, no. 4, pp. 691–695, 2015.

[13] D. Colley, C. Stanier, and M. Asaduzzaman, "Investigating the Effects of Object-Relational Impedance Mismatch on the Efficiency of Object-Relational Mapping Frameworks," *J. Database Manag.*, vol. 31, no. 4, 2020, doi: https://doi.org/10.4018/JDM.2020100101.

[14] J. Y. Seo, D. W. Lee, and H. M. Lee, "Performance comparison of CRUD operations in IoT based big data computing," 2017. doi: 10.18517/ijaseit.7.5.2674.

[15] H. A. Mumtahana, "Optimization of Transaction Database Design with MySQL and MongoDB," *Sink. J. dan Penelit. Tek. Inform.*, vol. 6, no. 3 SE-, pp. 883–890, Jul. 2022, doi: 10.33395/sinkron.v7i3.11528.

[16] S. Hamouda, Z. Zainol, and M. Anbar, "A Flexible Schema for Document Oriented Database (SDOD)," in *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019) - Volume 2: KEOD*, SciTePress, 2019, pp. 413–419. doi: 10.5220/0008353504130419.

[17] J. Forgaritenzo, A. Wibowo, and K. Wijana, "Pengembangan Program Alternatif untuk Proses Konsolidasi Multiple Database Menggunakan Pandas dan MongoDB ," *J. Sist. Komput. dan Inform.*, vol. 7, no. 1 SE-Articles, pp. 42–52, Sep. 2025, doi: 10.30865/json.v7i1.8320.

[18] R. Andreoli, T. Cucinotta, and D. Pedreschi, "RT-MongoDB: A NoSQL Database with Differentiated Performance," in *Proceedings of the 11th International Conference on Cloud Computing and Services Science - CLOSER*, SciTePress, 2021, pp. 77–86. doi: 10.5220/0010452400770086.