KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 6, No. 1, Januari (2025), pp. 197-209

Analisis Pengujian *Database* SQL dan NoSQL pada Aplikasi Berbasis *Microservice* dengan Spring Boot

Steven Arycena Fatich[1], Yeremia Alfa Susetyo[2]*
[1,2]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Salatiga, Jawa Tengah, Indonesia
E-mail: 672021189@student.uksw.edu[1], yeremia.alfa@uksw.edu[2]

Abstract

Efficient database utilization is essential in application development. This study aims to analyze the performance of PostgreSQL (SQL) and MongoDB (NoSQL) in CRUD operations and data aggregation using the Spring Boot framework. The research stages include developing a test application, conducting 10 simulation tests for each operation, and analyzing response times. The results indicate that MongoDB excels in create operations with an average response time of 230.4 ms and delete operations at 2.6 ms. In contrast, PostgreSQL demonstrates better efficiency in read operations and aggregations (sum, average, min, max) with superior response times.

Keywords: *SQL, NoSQL, PostgreSQL, MongoDB, Spring Boot, Response Time, Performance.*

Abstrak

Penggunaan database yang efisien menjadi tantangan utama dalam pengembangan sistem aplikasi. Penelitian ini bertujuan untuk menganalisis kinerja PostgreSQL (SQL) dan MongoDB (NoSQL) dalam menangani operasi CRUD dan agregasi data menggunakan framework Spring Boot. Tahapan penelitian meliputi pengembangan aplikasi uji, simulasi pengujian sebanyak 10 kali untuk setiap operasi, serta analisis hasil waktu respons. Hasil menunjukkan bahwa MongoDB unggul dalam operasi create dengan rata-rata waktu respons 230,4 ms dan delete sebesar 2,6 ms. Sebaliknya, PostgreSQL lebih efisien dalam operasi read serta agregasi (sum, average, min, max) dengan waktu respons yang lebih unggul.

Kata Kunci: *SQL, NoSQL, PostgreSQL, MongoDB, Spring Boot, Waktu Respons, Kinerja.*

1. Pendahuluan

Dalam menghadapi persaingan yang semakin ketat, perusahaan perlu meningkatkan layanan kepada *stakeholder* dengan menyediakan data terintegrasi yang memungkinkan pengambilan keputusan cepat dan akurat melalui proses penerapan *database* [1]. Di dunia bisnis, *database* memiliki peran yang penting dalam mendukung kegiatan operasional perusahaan yang lebih efektif dan efisien, serta menjadi komponen penting dari sistem informasi yang menyajikan informasi kepada perusahaan [2].

Database SQL dan *database* NoSQL memiliki pendekatan dan mekanisme yang berbeda dalam menyimpan dan mengelola data. *Database* SQL menyimpan data dalam struktur tabel yang terdiri dari baris dan kolom. Struktur ini sangat cocok untuk data yang terstruktur. Namun, karena menggunakan skema yang kaku dan tetap, *database* SQL kurang fleksibel dalam menangani data yang tidak terstruktur [3]. Di sisi lain, *database* NoSQL menawarkan fleksibilitas yang lebih besar dalam menangani data yang tidak terstruktur dan semi-terstruktur. NoSQL tidak menggunakan struktur tabel baris dan kolom. Sebaliknya, mereka menggunakan

ISSN: 2720-992X

model data seperti dokumen, graf, kolom, atau *key value*. Hal ini memungkinkan NoSQL untuk menyimpan dan mengelola data dalam format yang lebih alami dan sesuai dengan kebutuhan aplikasi. Namun, untuk mengolah data di *database* NoSQL, model data seringkali harus dirancang terlebih dahulu secara manual, yang bisa menjadi tantangan tambahan dalam perancangan dan pemeliharaan aplikasi [4].

Untuk mengintegrasikan data ke dalam *database*, *framework* Spring Boot dapat mempermudah proses integrasi data ke *database*. Spring Boot dapat dikonfigurasi dengan *dependencies* yang disediakan oleh Spring Boot. *Framework* ini bertanggung jawab untuk menampung, menyimpan, dan menganalisis data [5]. *Framework* Spring Boot memiliki *dependency* JPA (*Java Persistence API*) yang berfungsi untuk mengakses *database* SQL menggunakan anotasi pada *Model Class* [6]. Spring Boot menyediakan Spring Data MongoDB, sebuah *dependency* khusus untuk bekerja dengan *database* NoSQL seperti MongoDB. *Dependency* ini memungkinkan pengembang untuk mengakses dan mengelola data di MongoDB dengan mudah tanpa harus menulis *query* secara eksplisit, menggunakan pendekatan *repository* yang mirip dengan Spring Data JPA. Pengembang dapat memanfaatkan dua cara utama untuk berinteraksi dengan *database* MongoDB, yaitu menggunakan MongoTemplate atau MongoRepository. MongoTemplate memberikan kontrol yang lebih fleksibel, sementara MongoRepository menawarkan abstraksi yang lebih tinggi dan lebih mudah digunakan untuk operasi CRUD dasar [7]. Produk *database* yang dipilih adalah PostgreSQL sebagai perwakilan *database* SQL dan MongoDB sebagai perwakilan *database* NoSQL. *Response time* dapat dijadikan metode pengukuran dari *database* PostgreSQL dan MongoDB untuk melihat efisiensi waktu yang dibutuhkan dari kedua *database* [8].

Dari uraian di atas, *framework* Spring Boot menawarkan kelebihan *dependencies* yang mendukung untuk mengakses *database* baik SQL maupun NoSQL. Maka dari itu, *framework* Spring Boot memiliki peranan yang sangat penting untuk membandingkan efisiensi waktu dari *database* SQL yang diwakilkan oleh PostgreSQL dan *database* NoSQL yang diwakilkan oleh MongoDB.

2. Metodologi Penelitian

Pada penelitian yang berjudul “Pengukuran Kinerja Database SQL dan NoSQL Pada Aplikasi E-Commerce” disebutkan bahwa pilihan jenis *database* akan mempengaruhi potensi kompleksitas.

Penelitian ini bertujuan untuk menganalisis performa dan skalabilitas kinerja *database* SQL dan NoSQL yang digunakan dalam pengelolaan dan penyimpanan data untuk aplikasi *e-commerce*. Analisis ini dilakukan untuk menentukan *database* yang paling sesuai untuk aplikasi *e-commerce*, terutama dalam konteks masalah yang berkaitan dengan peningkatan beban kerja yang berlebihan, pertumbuhan data yang signifikan, dan kompleksitas transaksi data yang meningkat. Pengujian dilakukan secara komprehensif menggunakan pendekatan eksperimental yang melibatkan operasi baca dan tulis pada kedua jenis *database* dengan server yang sama. Indikator pengujian yang digunakan meliputi waktu eksekusi dan ukuran *database*. Hasil pengujian kemudian dianalisis secara objektif untuk membandingkan kinerja antara *database* SQL dan NoSQL berdasarkan *response time* [9].

Pada penelitian terdahulu menganalisis performa dan skalabilitas kinerja dari *database* SQL dan NoSQL yang digunakan dalam pengelolaan data untuk aplikasi *e-commerce* yang memiliki beban kerja tinggi, sehingga menjadi acuan dalam penelitian ini untuk menganalisis performa kinerja *database* SQL dan NoSQL dalam pengelolaan data perusahaan.

Pada penelitian yang berjudul "The Efficiency and Reliability of Backend Technologies: Express, Django, and Spring Boot" disebutkan framework Spring Boot memiliki efisiensi tertinggi dibandingkan framework Express dan Django karena framework Spring Boot menggunakan annotation dalam mengkonfigurasi data.

Penelitian ini membandingkan tiga *framework server-side* terkemuka, yaitu Django, Spring Boot, dan Express, dalam hal waktu pemrosesan permintaan dan keandalan, dengan hasil menunjukkan bahwa *framework* Spring Boot unggul dalam waktu pemrosesan dan keandalan, namun memiliki persentase kesalahan lebih tinggi ketika menjalankan beban tinggi dibandingkan dengan Express, sedangkan Django menunjukkan kinerja terburuk dalam hal waktu pemrosesan dan tingkat kesalahan [10].

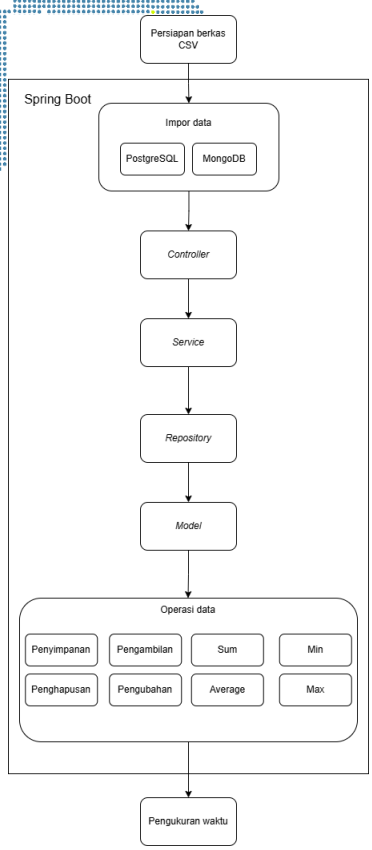
Penelitian terdahulu menyoroti perbandingan *framework* Spring Boot, Express, dan Django dengan hasil bahwa Spring Boot unggul dalam waktu pemrosesan permintaan GET, POST, PUT, dan DELETE. Spring Boot menunjukkan kecepatan waktu tanggapan rata-rata di bawah 1000 ms pada berbagai skenario permintaan. Berdasarkan data grafik, Spring Boot secara konsisten menunjukkan waktu pemrosesan permintaan yang lebih cepat dibandingkan dengan Express dan Django, terutama pada jumlah permintaan yang lebih tinggi (8.000 dan 16.000 permintaan). Pada 1.000 permintaan, Spring Boot memiliki waktu pemrosesan yang paling rendah, sedangkan pada 4.000 permintaan, waktu pemrosesan Spring Boot tetap lebih rendah, menunjukkan performa yang stabil dan cepat. Dalam hal keandalan, Spring Boot menunjukkan persentase permintaan yang salah tangani yang sangat rendah dibandingkan dengan Django dan Express, menjadikannya pilihan yang unggul untuk aplikasi yang membutuhkan performa tinggi dan keandalan dalam pemrosesan permintaan skala besar. Oleh karena itu, penelitian ini menggunakan *framework* Spring Boot dalam analisis perbandingan kinerja *database* SQL dan NoSQL.

Penelitian yang berjudul "Perbandingan Waktu Respon Aplikasi Database NoSQL Elasticsearch dan MongoDB pada Pengujian Operasi CRUD" disebutkan bahwa MongoDB unggul dalam semua model pengujian DML yang terdiri dari proses memasukkan data, membaca data, *update* data, dan hapus data. Namun memiliki kelemahan pada proses *query select* dibandingkan dengan MySQL PHP 7.2.27.

Penelitian terdahulu melakukan pengujian perbandingan berdasarkan *response time*, aplikasi *database* NoSQL MongoDB menunjukkan kinerja yang lebih cepat dibandingkan dengan aplikasi *database* NoSQL Elasticsearch dalam semua operasi CRUD dengan perbedaan yang signifikan [11].

Penelitian terdahulu menunjukkan kinerja *database* NoSQL MongoDB lebih cepat dibandingkan *database* NoSQL Elasticsearch sehingga menjadi acuan penelitian ini untuk membandingkan *database* SQL yang diwakilkan oleh PostgreSQL dengan *database* NoSQL MongoDB menggunakan *framework* Spring Boot.

Pada penelitian ini, metode yang digunakan melibatkan tahapan integrasi data menggunakan *framework* Spring Boot yang dirancang untuk mengukur performa *database* PostgreSQL dan MongoDB dalam operasi *create*, *read*, *update*, *delete*, *sum*, *average*, *min*, dan *max*.



Gambar 1. *Flowchart* Aplikasi Pengujian Kinerja *Database*

Berdasarkan Gambar 1, tahap awal adalah persiapan berkas CSV yang akan digunakan dalam penelitian sebagai sumber data. Proses ini melibatkan pengumpulan dataset yang relevan dan memastikan format data sesuai dengan kebutuhan sistem. Data CSV yang sudah disiapkan akan menjadi sumber data yang akan diimpor ke dalam *database* PostgreSQL dan MongoDB pada tahap selanjutnya.

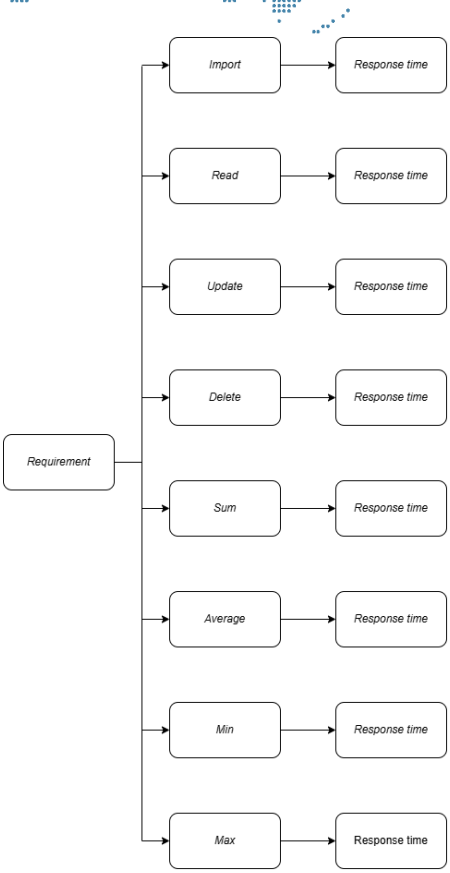
Tahap berikutnya melibatkan proses impor data dari *file* CSV ke dalam dua *database* yang berbeda, yaitu PostgreSQL dan MongoDB menggunakan *framework* Spring Boot. Proses ini dilakukan dengan *Controller* yang menerima permintaan impor data. Dengan cara ini, data yang diimpor ke kedua *database* adalah data yang identik, memungkinkan perbandingan performa secara konsisten pada tahap-tahap selanjutnya.

Setelah melalui *Controller*, data akan diproses oleh *Service*. Pada tahap ini, *service* berperan sebagai penghubung antara *Controller* dan *Repository*, memastikan setiap data dari *file* CSV diteruskan dengan benar ke *Repository* yang sesuai berdasarkan *database* yang dituju.

Data kemudian disimpan dalam database menggunakan Repository yang sesuai dengan masing-masing database. *Repository* bertanggung jawab untuk mengelola operasi penyimpanan data ke dalam *database* dan mengkases *Model* yang telah dirancang untuk konsistensi struktur data. *Model* ini akan menyesuaikan setiap kolom dan dokumen dari data yang diimpor, sehingga kedua *database* memiliki salinan data yang identik.

Pada tahap operasi data, dilakukan operasi *create*, *read*, *update*, *delete*, *sum*, *average*, *min*, dan *max*. Setiap operasi dilakukan secara bersamaan pada PostgreSQL dan MongoDB untuk memastikan kesesuaian dan memungkinkan perbandingan performa.

Tahap terakhir adalah pengukuran waktu, di mana waktu yang dibutuhkan untuk setiap operasi pada PostgreSQL dan MongoDB dihitung tepat sebelum operasi dijalankan dan setelah operasi selesai. Pengukuran waktu ini memberikan gambaran komparatif mengenai performa dari kedua *database* dalam menangani operasi yang sama.



Gambar 2. *Incremental Model*

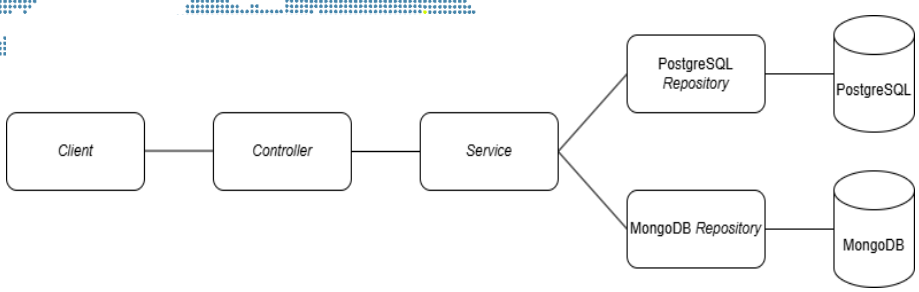
Pada tahap ini, kebutuhan sistem diidentifikasi dan dirumuskan. Proses ini melibatkan memahami persyaratan pengguna untuk mengelola data yang diimpor dari *file* CSV ke dua jenis *database* yang berbeda, yaitu PostgreSQL dan MongoDB.

Pada tahap implementasi, kode yang mendukung fungsionalitas yang telah dirancang dikembangkan. Setiap *Service* dan *Controller* diimplementasikan sesuai dengan desain yang telah dibuat. Tahap ini melibatkan penulisan kode sumber dan integrasi komponen-komponen sistem.

Tahap pengujian memastikan bahwa setiap komponen sistem bekerja dengan baik dan sesuai dengan spesifikasi yang telah ditetapkan. Service operasi *create*, *read*, *update*, *delete*, *sum*, *average*, *min*, dan *max* diuji di kedua *database*. Ketika dilakukan pengujian, pada Postman ditampilkan waktu yang diperlukan dari kedua *database*.

3. Hasil dan Pembahasan

Hasil *response time* yang diperoleh pada setiap operasi *create*, *read*, *update*, *delete*, *sum*, *average*, *min*, dan *max* akan menunjukkan performa kinerja dari MongoDB sebagai perwakilan dari *database* NoSQL dan PostgreSQL sebagai perwakilan dari *database* SQL.



Gambar 3. Arsitektur Diagram

Pada proses persiapan membaca semua *file* dengan format *Comma Seperated Value* (CSV) yang sudah disimpan pada *local directory*. Lalu setelah membaca semua data yang berjumlah 116 *file* format CSV dengan total baris sejumlah 3419 baris, maka *database* siap diukur berdasarkan *response time*.

Kode program 1 Kode Program untuk Membaca file dengan format CSV

```
…
(FileReader fileReader = new FileReader(file);
            CSVReader csvReader = new CSVReaderBuilder(fileReader).withSkipLines(0).build())
{
        String[] headers = csvReader.readNext();
        String[] values;

        while ((values = csvReader.readNext()) != null) {
            PostgresData postgresData = new PostgresData();
            MongoData mongoData = new MongoData();
            for (int i = 0; i < headers.length; i++) {
                String fieldName = headers[i];
                String fieldValue = values[i];
…
            postgresDataList.add(postgresData);
            mongoDataList.add(mongoData);
        }
    } catch (Exception e) {
        return durations;
    }

    Instant start = Instant.now();
    postgresDataRepository.saveAll(postgresDataList);
    Instant postgresEnd = Instant.now();

    mongoDataRepository.saveAll(mongoDataList);
    Instant mongoEnd = Instant.now();

    Duration postgresDuration = Duration.between(start, postgresEnd);
    Duration mongoDuration = Duration.between(postgresEnd, mongoEnd);

    durations.put("PostgreSQL Time (ms)", postgresDuration.toMillis());
    durations.put("MongoDB Time (ms)", mongoDuration.toMillis());
…
```

Kode program 1 menunjukkan program untuk membaca *file* CSV menggunakan ‘FileReader’ dan ‘CSVReader’. ‘FileReader’ bertugas untuk membuka *file* yang akan diproses, sementara ‘CSVReader’ bertugas membantu membaca data dari *file* CSV setiap baris. Fungsi ‘.withSkipLines(0).build()’ digunakan untuk

menginisialisasi ‘CSVReader’ *membaca* semua baris dari *file* CSV. Setelah itu, ‘headers’ mengambil baris pertama *file* CSV yang berisi nama kolom, sedangkan ‘value’ membaca setiap baris data yang akan diolah. Dengan menggunakan iterasi ‘while’, setiap baris *file* CSV akan diakses secara berurutan, dan setiap nilai dalam baris disesuaikan dengan nama kolomnya. Hal ini memastikan bahwa setiap data dari kolom *file* CSV diolah dengan benar ke dalam objek yang sesuai.

Pada setiap iterasi ‘while’, dua objek ‘PostgresData’ dan ‘MongoData’, dibuat untuk menampung data dari baris yang sedang dibaca. Di dalam iterasi ‘for’, setiap kolom pada baris tersebut dipetakan ke *field* yang relevan di ‘PostgresData’ dan ‘MongoData’ berdasarkan nama kolom. Sebagai contoh, jika ‘fieldName’ adalah "COMPANY_CODE", maka nilai ‘fieldValue’ dari kolom tersebut akan dimasukkan ke *field* COMPANY_CODE pada kedua objek ‘PostgresData’ dan ‘MongoData’. Hal ini dilakukan untuk memastikan bahwa setiap baris *file* CSV dipetakan dengan benar ke struktur data dalam aplikasi.

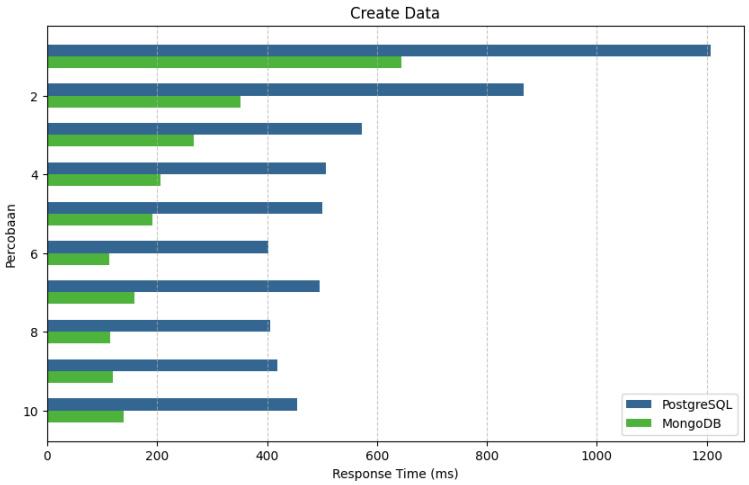
Setelah setiap baris diolah menjadi objek ‘PostgresData’ dan ‘MongoData’, kedua objek tersebut ditambahkan ke dalam ‘postgresDataList’ dan ‘mongoDataList’. Kedua *List* ini akan digunakan untuk menyimpan semua data dari *file* CSV sebelum disimpan ke dalam *database*. Proses ini membantu memastikan bahwa data yang sama tersedia dalam dua format berbeda yang sesuai untuk PostgreSQL dan MongoDB.

Setelah berhasil *parsing* seluruh data dari CSV, data dalam ‘postgresDataList’ disimpan ke dalam *database* PostgreSQL menggunakan metode ‘postgresDataRepository.saveAll(postgresDataList)’, diikuti dengan pengukuran waktu yang diperlukan untuk operasi ini. Setelah penyimpanan ke PostgreSQL selesai, data dari ‘mongoDataList’ disimpan ke MongoDB menggunakan ‘mongoDataRepository.saveAll(mongoDataList)’, dan waktu penyimpanan juga diukur. Pengukuran waktu untuk kedua operasi ini menggunakan ‘Instant’ untuk menandai waktu mulai dan waktu selesai. *Durasi* masing-masing operasi dihitung menggunakan ‘Duration.between()’, dan hasilnya dimasukkan ke dalam ‘durations’ untuk mencatat waktu yang diperlukan oleh PostgreSQL dan MongoDB.

Struktur data ‘PostgresData’ dan ‘MongoData’ masing-masing digunakan untuk menyimpan data yang cocok dengan PostgreSQL dan MongoDB. Meskipun keduanya menampung data yang sama dari *file* CSV, perbedaannya terletak pada bagaimana data disimpan dan diakses dalam kedua *database*, yang akan dibandingkan dalam penelitian ini. Proses pengukuran waktu pada penyimpanan data ini menjadi salah satu parameter penting dalam penelitian, memberikan gambaran mengenai performa kedua *database*. Dengan mengukur waktu yang diperlukan untuk operasi penyimpanan di masing-masing *database*.

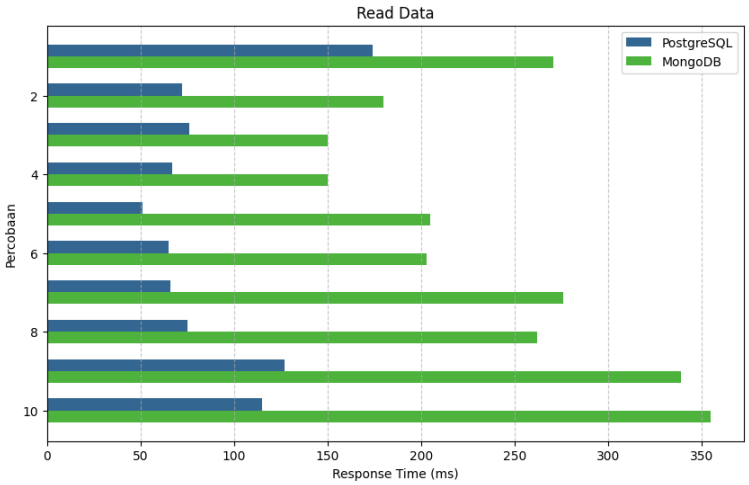
Gambar 4. Hasil Pengujian

Pada gambar 4, respons JSON yang diterima menunjukkan hasil pengujian waktu pemrosesan yang dibutuhkan untuk satu kali operasi impor dari masing-masing *database*. Terdapat total waktu yang dibutuhkan dari kedua *database*, jumlah *file* yang diproses, serta rata-rata waktu respons yang dibutuhkan dari kedua *database*.



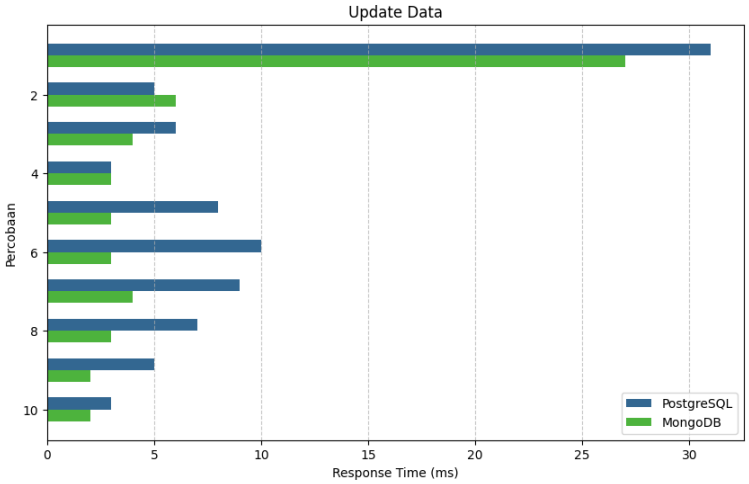
Gambar 5. Hasil *create* data

Gambar 5 menunjukkan bahwa MongoDB konsisten memiliki waktu respons yang lebih singkat dibandingkan PostgreSQL pada operasi *create* data yang diulang sebanyak 10 kali. Hal ini dikarenakan MongoDB mendukung penyimpanan dokumen secara fleksibel tanpa harus mendefinisikan skema tabel secara eksplisit seperti pada PostgreSQL atau *relational database* lainnya. Fleksibilitas ini mengurangi *overhead* ketika harus menyesuaikan data dengan skema yang ketat, sehingga mempercepat proses *create* data [12].



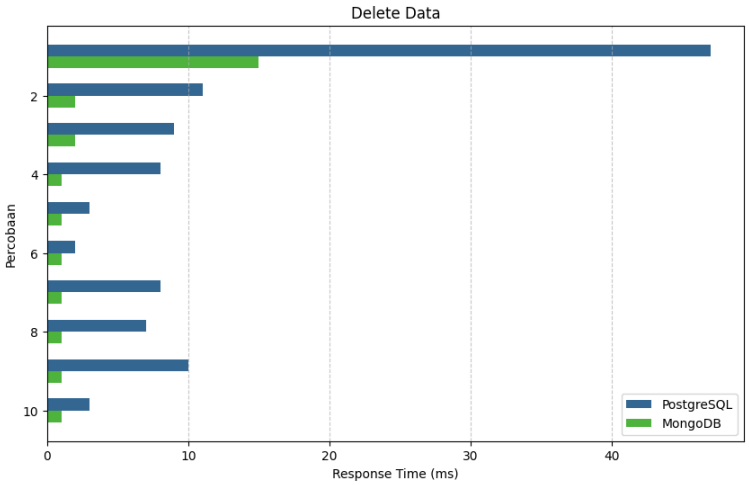
Gambar 6. Hasil *read* data

Pada Gambar 6, proses *read* data berbanding terbalik dengan PostgreSQL yang unggul dibanding MongoDB pada 10 kali percobaan. Kinerja unggul PostgreSQL pada proses *read* data karena dipengaruhi oleh kemampuan *indexing* yang efektif dalam mengurangi waktu respons serta memiliki ukuran *dataset* yang lebih kecil dalam sistem *database* [13].



Gambar 7. Hasil *update* data

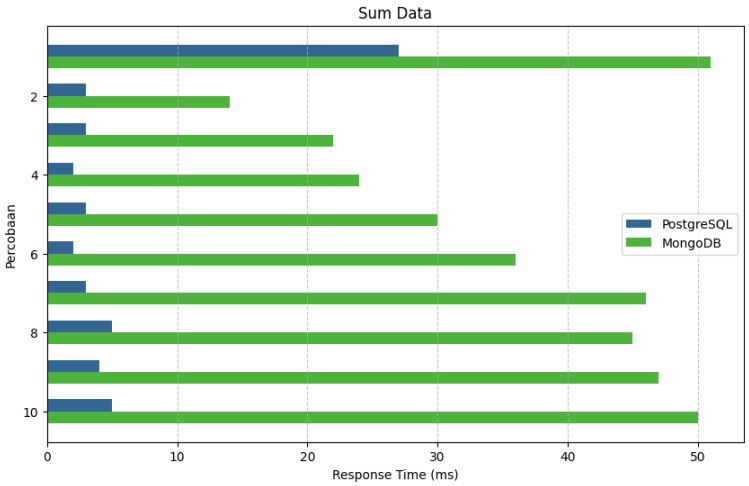
Gambar 7 menggambarkan proses *update* data untuk satu baris dan dokumen secara bersamaan menggunakan ‘id’ di masing-masing *database*. Pada 10 kali percobaan, PostgeSQL sempat mengungguli waktu respons pada percobaan ke dua dan mencatat hasil yang sama pada percobaan ke empat. Namun, secara keseluruhan MongoDB tetap unggul karena struktur data dokumen yang fleksibel pada MongoDB memungkinkan pembaruan data secara *granular* tanpa perlu mengubah keseluruhan struktur data. Selain itu, MongoDB mendukung *indexing* sekunder pada berbagai *field*, memungkinkan pencarian dan pembaruan data yang cepat berdasarkan indeks tersebut. Meskipun PostgreSQL juga mendukung *indexing*, namun mekanismenya tidak seefisien MongoDB dalam menangani operasi *write* dengan beban kerja yang tinggi [14].



Gambar 8. Hasil *delete* data

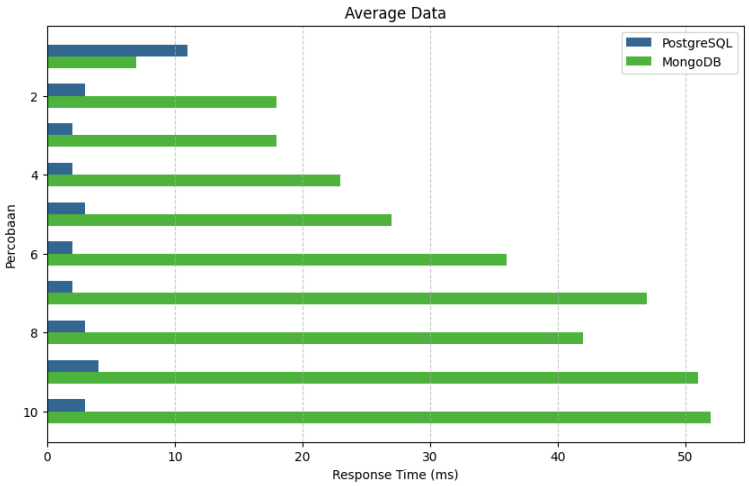
Pada Gambar 8, proses *delete* data satu baris dan dokumen secara bersamaan menggunakan ‘id’ di masing-masing *database*. MongoDB tercatat lebih cepat pada seluruh percobaan dibandingkan dengan PostgreSQL. Hal ini dikarenakan MongoDB mendukung *granular deletion* yang memungkinkan operasi *delete* dieksekusi dengan presisi pada dokumen yang spesifik. Pendekatan menggunakan *granular* ini mengurangi dampak pada dokumen data yang lain dan meningkatkan efisiensi. Di sisi lain, PostgreSQL membutuhkan *query* yang lebih kompleks untuk

mengeksekusi operasi *delete* pada baris yang spesifik, yang memungkinkan PostgreSQL membutuhkan waktu respons lebih lama dibandingkan MongoDB [15].



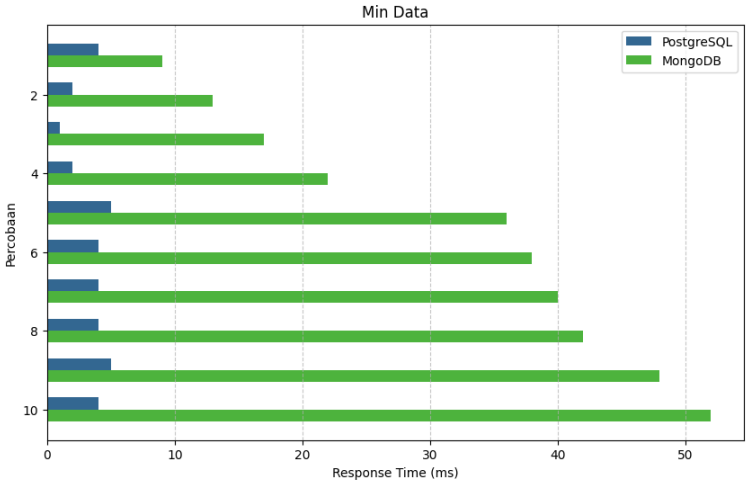
Gambar 9. Hasil *sum* data

Gambar 9 memperlihatkan operasi *sum* pada kolom "Total_Net_Amount" yang berisi tipe data *Double*. PostgreSQL mengungguli MongoDB pada 10 kali percobaan dengan hasil yang signifikan. PostgreSQL menggunakan model data relasional yang terstruktur, hal ini memungkinkan optimasi yang lebih baik untuk operasi agregasi. Selain itu, PostgreSQL memiliki dukungan berbagai tipe data yang kuat, termasuk tipe data numerik yang memiliki tingkat kepresisian tinggi. Meskipun model data dokumen MongoDB bersifat fleksibel, model data dokumen ini kurang optimal untuk mengeksekusi operasi agregasi terutama yang melibatkan perhitungan dengan banyak data [16].



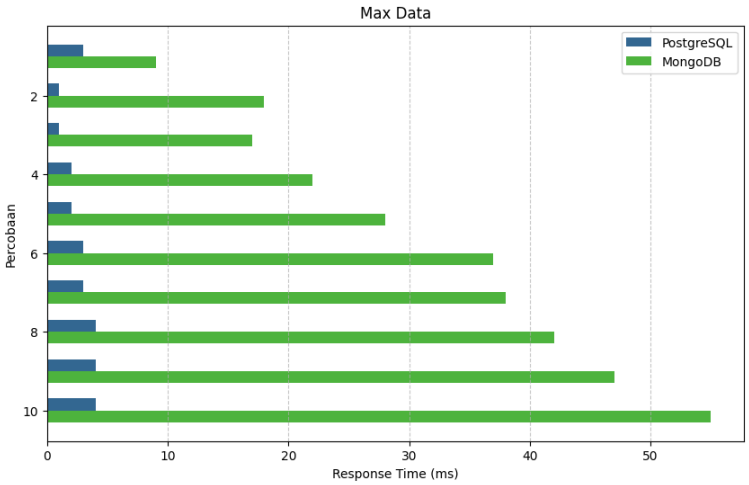
Gambar 10. Hasil *average* data

Pada gambar 10 menunjukkan bahwa MongoDB lebih cepat dalam melakukan operasi average pada kolom "Total_Net_Amount" pada percobaan pertama, namun pada percobaan lainnya mencatat waktu yang lebih lambat dibandingkan dengan PostgreSQL. PostgreSQL dapat mencatat waktu respons yang lebih cepat secara konsisten karena kemampuan *caching*, PostgreSQL dapat memberikan kinerja yang jauh lebih baik terutama untuk *query* yang sering dijalankan [17].



Gambar 11. Hasil *min* data

Pada Gambar 11, operasi untuk mencari nilai minimum (*min*) pada kolom "Total_Net_Amount" di MongoDB mencatat waktu yang lebih lambat dibandingkan dengan PostgreSQL pada seluruh percobaan dengan selisih yang cukup signifikan. Hal ini dikarenakan PostgreSQL memiliki optimasi *query* yang lebih matang dan mampu menghasilkan rencana eksekusi yang lebih efisien, salah satunya untuk operasi agregasi mencari nilai minimum [18].



Gambar 12. Hasil *max* data

Gambar 12 menunjukkan bahwa operasi untuk mencari nilai maksimum (*max*) pada kolom "Total_Net_Amount" di MongoDB mencatat *response time* yang lebih lambat ketika dilakukan percobaan 10 kali ketimbang PostgreSQL yang mencatat hasil lebih cepat serta lebih konsisten. PostgreSQL dapat memberikan performa yang konsisten dan lebih cepat karena PostgreSQL memiliki dukungan agregasi yang baik. Fungsi-fungsi agregasi seperti *max* telah dioptimalkan pada PostgreSQL [19].

4. Kesimpulan

Hasil percobaan menunjukkan bahwa performa PostgreSQL dan MongoDB bervariasi tergantung pada jenis operasi yang dilakukan. Pada operasi *create*, MongoDB mencatatkan rata-rata waktu respons 230,4 ms, yang lebih cepat

dibandingkan PostgreSQL dengan rata-rata 582,9 ms. Pada operasi *delete*, MongoDB juga unggul dengan rata-rata waktu respons 2,6 ms, jauh lebih cepat daripada PostgreSQL yang mencatatkan 10,8 ms. Namun, pada operasi *read*, PostgreSQL menunjukkan hasil yang lebih cepat dengan rata-rata waktu respons 88,8 ms dibandingkan MongoDB dengan 239,1 ms di seluruh percobaan. Pada proses *update*, PostgreSQL sempat unggul pada percobaan pertama, tetapi secara keseluruhan MongoDB lebih cepat dengan rata-rata waktu respons 5,7 ms, dibandingkan PostgreSQL yang mencatatkan 8,7 ms. Untuk operasi agregasi seperti *sum*, *min*, dan *max*, PostgreSQL lebih unggul di seluruh percobaan dengan rata-rata waktu respons masing-masing 5,7 ms, 3,5 ms, dan 2,7 ms dibandingkan MongoDB yang mencatatkan waktu 36,5 ms, 31,7 ms, dan 31,3 ms. Pada operasi *average*, PostgreSQL mencatatkan hasil yang lebih konsisten dengan rata-rata waktu 3,5 ms, sementara MongoDB unggul di percobaan pertama tetapi memiliki rata-rata lebih tinggi, yakni 32,1 ms. Secara keseluruhan, MongoDB lebih unggul dalam operasi manipulasi data individu seperti *create* dan *delete*. Di sisi lain, PostgreSQL menunjukkan performa yang lebih optimal untuk operasi pembacaan data dan analitis seperti *read*, *sum*, *average*, *min*, dan *max*, menjadikannya lebih ideal untuk *query* yang lebih kompleks.

Penelitian selanjutnya disarankan untuk memperluas eksperimen ini dengan *dataset* yang lebih besar dan variasi struktur data untuk melihat apakah perbedaan performa antara PostgreSQL dan MongoDB tetap konsisten. Selain itu, menggunakan indeks dan optimasi konfigurasi khusus pada kedua *database* juga bisa dieksplorasi untuk melihat dampaknya terhadap performa. Penelitian juga bisa melibatkan aspek skalabilitas dan responsivitas pada lingkungan sistem yang lebih besar atau dengan beban akses yang lebih tinggi untuk mendapatkan gambaran yang lebih komprehensif mengenai perbedaan performa kedua *database* ini dalam skenario data perusahaan yang sesungguhnya.

Daftar Pustaka

[1] D. M. Sagala, L. Rahmadani, Y. Rahmadani, E. S. Wahyuningsih, A. Arifah Dan N. F. Lawita, "Penerapan Database Pada Perusahaan (Studi Penerapan Erp Pada Pt. Sinar Sosro)," *Jurnal Pendidikan Tambusai*, Vol. 5, No. 2, Pp. 1-2, 2021.

[2] S. T. Nurhayati Dan M. I. Nasution, "Database Management System Pada Perusahaan," *Jurnal Akuntansi Keuangan Dan Bisnis*, Vol. 1, No. 2, Pp. 62-64, 2023.

[3] A. Fadli, M. I. Zulfa, A. W. W. Nugraha, A. Taryana Dan M. S. Aliim, "Analisis Perbandingan Unjuk Kerja Database Sql Dan Database Nosql Untuk Mendukung Era Big Data," *Jurnal Nasional Teknik Elektro*, Vol. 9, No. 3, Pp. 155-156, 2020.

[4] A. S. Maharani, C. F. Alifa Dan A. L. Febriga, "Perancangan Data Base Kasir Dan Persediaan Barang Menggunakan Mongodb," *Jdmsi*, Vol. 3, No. 1, Pp. 32-33, 2022.

[5] F. Zhang, G. Sun, B. Zheng Dan L. Dong, "Design And Implementation Of Energy Management System Based On Spring Boot Framework," *Information*, Vol. 12, No. 457, Pp. 1-2, 2021.

[6] W. C. U. Dagha Dan Y. A. Susetyo, "Pembangunan Aplikasi Web Event Menggunakan Framework Spring Boot Di Pt Xyz," *Jurnal Teknik Informatika Dan Sistem Informasi*, Vol. 8, No. 3, Pp. 2-3, 2021.

[7] D. Vohra, "Using Spring Data With Mongodb," Dalam *Buku "Pro Mongodb™ Development"*, New York, Apress, 2020, Pp. 2-3.

[8] M. F. Abdi, A. Susanto Dan Kusnawi, "Perbandingan Kecepatan Pencarian Data

Sql Dan Nosql," *Jurnal Teknologi Informasi*, Vol. 5, No. 1, Pp. 2-3, 2021.

[9] M. S. D. Pujas, M. I. Amar, M. E. Arif Dan M. M. Mustamin, "Pengukuran Kinerja Database Sql Dan Nosql Pada Aplikasi E-Commerce," *Jurnal Fokus Elektroda*, Vol. 9, No. 1, Pp. 1-2, 2024.

[10] D. Choma, K. Chwaleba Dan M. Dzienkowski, "The Efficiency And Reliability Of Backend Technologies: Express, Django, And Spring Boot," *Iapgoś,* Vol. 4, No. 4, Pp. 3-5, 2023.

[11] T. L. Sinaga, N. Charibaldi Dan N. H. Cahyana, "Perbandingan Waktu Respon Aplikasi Database Nosql Elasticsearch Dan Mongodb Pada Pengujian Operasi Crud," *Jiska (Jurnal Informatika Sunan Kalijaga),* Vol. 8, No. 1, Pp. 9-13, 2023.

[12] S. Keshavarz, "Analyzing Performance Differences Between Mysql And Mongodb," Technology Arts Sciences Th Köln, Koln, 2021.

[13] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis Dan D. Anagnostopoulos, "Mongodb Vs Postgresql: A Comparative Study On Performance Aspects," *Geoinformatica,* Vol. 25, Pp. 264-265, 2020.

[14] I. Dalström Dan P. Ericsson, "Performance Comparison Between Postgresql, Mongodb, Arangodb And Hbase," *Bachelor Degree Project In Information Technology,* Pp. 19-36, 2022.

[15] A. Vatjalainen, "Sql Versus Nosql," *Hämeen Ammatti-Korkeakoulu,* Pp. 24-30, 2023.

[16] M. Q. Porter-Brown, "Accelerating Aggregation Efficiency: Using Postgres As A Cache With Mongodb," *Bard Digital Commons,* Pp. 46-48, 2021.

[17] M. Q. Porter-Brown, "Accelerating Aggregation Efficiency: Using Postgres As A Cache With Mongodb," *Bard Digital Commons,* Pp. 49-51, 2021.

[18] L. Alves, P. Oliveira, J. Rocha, C. Wanzeller, F. Cardoso, P. Martins Dan M. Abbasi, "Comparison Of Semi-Structured Data On Mssql And Postgresql," *Marketing And Smart Technologies,* Pp. 11-12, 2023.

[19] D. Joiner, M. Clement, S. (. Chan, K. Pereira, A. Wong, Y. Khmelevsky, J. Mahony Dan M. Ferri, "Dw Vs Oltp Performance Optimization In The Cloud On Postgresql (A Case Study)," *2022 Ieee International Conference On Recent Advances In Systems Science And Engineering (Rasse),* Pp. 6-7, 2022.