Analisis Performa *Load Testing* Antara *Mysql* Dan *Nosql Mongodb* Pada *RestAPI Nodejs* Menggunakan *Postman*

Lifan Dwinur Andrianto[1], Dwi Fatrianto Suyatno[2].

[1,2] Sistem Informasi, Fakultas Teknik, Universitas Negeri Surabaya
[1]lifan.19026@mhs.unesa.ac.id
[2]dwifatrianto@unesa.ac.id

***Abstrak*— Dengan perkembangan teknologi yang cepat, terutama di bidang industri dan penggunaan internet yang meluas di Indonesia, media penyimpanan data yang efektif diperlukan. Seperti *MySQL*, *database* manajemen sistem (*DBMS*) telah menjadi solusi yang efektif untuk mengelola dan memproses data. Sebagai jenis *DBMS*, *MySQL* telah menunjukkan kemampuan untuk mendukung pengembangan aplikasi oleh pengguna yang berpengalaman maupun pemula. Di sisi lain, dalam era digital yang terus berkembang, aplikasi *web* dan *mobile* menjadi penting, dan *Node.js* dan *REST API* menjadi dasar untuk pembuatan aplikasi *web* yang efektif. Kemajuan teknologi juga menghasilkan alternatif penyimpanan data tanpa relasi, yang dikenal sebagai *NoSQL*. *MongoDB*, misalnya, menawarkan fleksibilitas dan kinerja yang cepat saat menyimpan data dalam format dokumen *JSON*. Analisis performa pengujian load antara *MySQL* dan *MongoDB* pada aplikasi *Node.js* sangat penting karena keduanya memiliki model data dan jenis beban kerja yang berbeda. Ini adalah teknik penting untuk mengukur dan mengevaluasi performa aplikasi. Kecepatan pencarian, penyimpanan data, dan kompleksitas *query* dipengaruhi oleh model data relasional *MySQL* dan *MongoDB NoSQL*. Tujuan dari analisis performa *load testing* ini adalah untuk mendapatkan pemahaman tentang bagaimana *MySQL* dan *MongoDB* berkinerja saat dihadapkan pada beban kerja yang berbeda, khususnya dalam konteks aplikasi *Node.js*. Hasil analisis ini diharapkan memberikan gambaran yang jelas tentang kelebihan dan kekurangan masing-masing *database* saat digunakan dengan aplikasi *Node.js*, sehingga pengembang dan arsitek sistem dapat membuat keputusan yang bijaksana dan berdasarkan data ketika mereka menggunakan aplikasi *Node.js*.**

***Kata Kunci*— *MySQL*, *MongoDB*, *RestAPI*, *NodeJS*, *Load* Testing.**

I. *PENDAHULUAN*

Kemajuan teknologi yang begitu pesat memiliki banyak manfaat bagi masyarakat dalam berbagai bidang. Pada bidang industri, kemajuan teknologi dapat membantu perusahaan dalam meningkatkan produksi mereka [1]. Hasil survei APJII (Asosiasi Penyelenggara Jasa Internet Indonesia) tahun 2019 dan 2020 menunjukkan bahwa pengguna internet di Indonesia kurang lebih mencapai 73,7% dari total populasi [2]. Untuk melakukan pengolahan data, dibutuhkan media penyimpanan yang baik. Tentu saja, penggunaan Database Management System (DBMS) sebagai mengandalikan pembuatan, pengolahan, dan pemeliharaan data yang efektif dan efisien adalah sesuatu yang mendukung [3].

MySQL merupakan suatu jenis *database* server yang sangat terkenal. *MySQL* termasuk jenis *RDBMS* (*Relational Database Manajement System*) [4]. *RDBMS* adalah program yang memungkinkan pengguna *database* untuk membuat, mengelola, dan menggunakan data pada suatu model relational [5]. *MySQL* memiliki kemampuan cukup baik untuk menunjang kerja para *developer*, baik user yang sudah berpengalaman dengan *database* maupun untuk pemula. *MySQL* menggunakan bahasa *SQL* untuk mengakses *database*-nya [6]. Lisensi *MySQL* adalah pengecualian untuk lisensi perangkat lunak sumber terbuka dan juga memiliki versi komersial. Tag *Mysql* adalah "*Database Open Source* Paling Populer di Dunia.".

NoSQL menggunakan nilai kunci pada *database* program dan menggunakan nilai kunci untuk mengidentifikasi data. Ini berarti bahwa *NoSQL* tidak membutuhkan hubungan data atau pertanyaan untuk melakukannya. Jadi, *NoSQL* sangat berbeda dari *DBMS*. Salah satu contohnya adalah *MongoDB* [7], yang menawarkan kinerja *database* yang cepat dan efisien dibandingkan dengan jenis *database* lain. Dokumen yang digunakan dalam *database* ini diformat dalam format *JSON* yang lebih mudah digunakan. *MongoDB* memiliki kemampuan untuk menyimpan jumlah data yang lebih besar karena memiliki *memori cache*..

Dalam era digital yang terus berkembang, aplikasi *web* dan *mobile* menjadi bagian integral dari kehidupan sehari-hari. *Rest API (Application Programming Interface)* telah menjadi fondasi utama untuk komunikasi antara berbagai aplikasi, dan *Node.js* menjadi salah satu lingkungan yang sangat populer untuk mengembangkan *server-side* dari aplikasi *web* yang efisien dan *scalable* [8].

Load Testing merupakan teknik *performance testing* dimana respon sistem diukur dalam berbagai kondisi dan beban. Pengujian ini membantu menentukan bagaimana *software* berperilaku ketika beberapa *user* mengakses *software* secara bersamaan [9]. *Load testing* merupakan suatu pendekatan yang kritis dalam mengukur dan mengevaluasi performa suatu aplikasi. Dalam konteks analisis performa *load testing*, penting untuk memahami bagaimana *MySQL* dan *MongoDB* bertindak sebagai penyimpanan data untuk aplikasi *Node.js*, terutama saat dihadapkan pada beban kerja yang tinggi [10].

Ada beberapa alasan mengapa perbandingan performa antara *database MySQL* dan *MongoDB* penting, terutama dalam konteks pengembangan aplikasi *Node.js* diantaranya

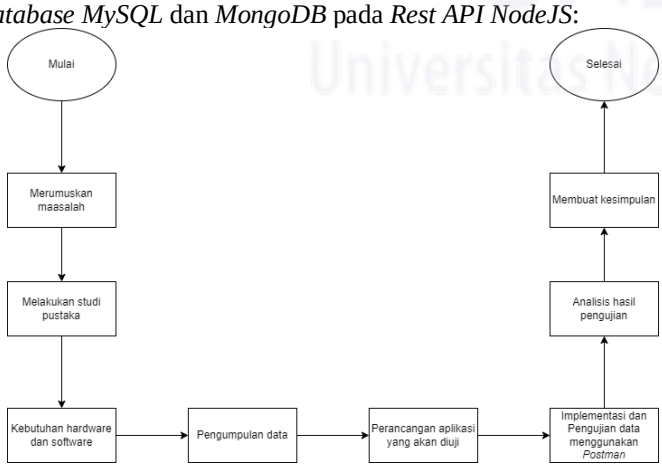
adalah sebagai berikut. Aspek Model Data *MySQL* merupakan *database* relasional yang menggunakan skema tabel dengan relasi antar tabel. *MongoDB*, di sisi lain, adalah *database NoSQL* yang menyimpan data dalam format dokumen *JSON* yang fleksibel. Perbedaan dalam model data ini dapat berdampak pada kinerja, terutama dalam hal kecepatan pencarian, penyimpanan data, dan kompleksitas *query*. Serta aspek Beban Kerja Spesifik, Performa *database* dapat dipengaruhi oleh jenis beban kerja yang dihadapi. *MySQL* mungkin lebih cocok untuk aplikasi dengan *query* kompleks dan hubungan yang kuat antar data, sementara *MongoDB* mungkin lebih unggul dalam situasi di mana skema data lebih dinamis dan aplikasi memiliki kebutuhan untuk menangani dokumen dengan cepat.

Tujuan dari analisis ini adalah untuk memahami bagaimana performa *load testing* dari dua *database* tersebut mempengaruhi response dan kinerja aplikasi *Node.js* pada tingkat beban yang berbeda. Dengan membandingkan performa *MySQL* dan *MongoDB*, kita dapat mengevaluasi kelebihan dan kekurangan masing-masing dalam konteks penggunaan dengan aplikasi *Node.js*. Hasil dari analisis performa *load testing* ini dapat memberikan pandangan yang lebih jelas terkait kebutuhan dan preferensi dalam pemilihan *database* untuk aplikasi *Node.js*, memungkinkan pengembang dan arsitek sistem untuk membuat keputusan yang informasional dan berdasarkan data. Dengan demikian, analisis ini dapat membantu meningkatkan skalabilitas, keandalan, dan efisiensi aplikasi yang dibangun dengan menggunakan kombinasi *Node.js*, *MySQL*, dan *MongoDB*. Berdasarkan latar belakang diatas, maka penulis mengajukan penelitian dengan judul "ANALISIS PERFORMA *LOAD TESTING* ANTARA *MYSQL* DAN *NOSQL MONGODB* PADA *RESTAPI NODEJS* MENGGUNAKAN *POSTMAN*".

II. METODOLOGI PENELITIAN

A. Alur Penelitian

Berikut merupakan alur penelitian yang akan dilakukan oleh peneliti untuk melakukan pengujian *load testing database MySQL* dan *MongoDB* pada *Rest API NodeJS*:



Gbr. 1 *Flowchart* Alur Penelitian.

Berikut adalah penjelasan mengenai alur diatas:

1) Merumuskan masalah sesuai dengan yang tertulis pada latar belakang penelitian.
2) Peneliti melakukan studi pustaka guna mencari referensi teori yang akan diuji.
3) Mengidentifikasi *hardware* dan *software* yang akan digunakan untuk melakukan pengujian.
4) Setelah itu mengumpulkan data yang akan digunakan untuk aplikasi yang diuji nantinya.
5) Selanjutnya membuat aplikasi yang akan diuji dengan menggunakan *rest-api nodejs*.
6) Melakukan testing dengan *postman* pada kedua *database* yang sudah terkonfigurasi dengan *rest-api* yang sudah dirancang.
7) Setelah hasil dari *load test* sudah ada, selanjutnya adalah menganalisis serta membandingkan hasil dari pengujian sebelumnya.
8) Terakhir, peneliti akan membuat rangkuman ke dalam kesimpulan sebagai kajian akhir peneliti.

B. Rumusan Masalah

Berdasarkan latar belakang diatas maka tujuan penelitian ini bermaksud melakukan pengujian *load testing API* dari dua arsitektur *database* yang berbeda dan memberikan analisa *load testing API* sebagai refrensi untuk menentukan manakah *database* yang tepat saat akan melakukan proses *development* aplikasi. Pengujian performa dengan metode *load testing* berfokus pada 4 indikator yaitu *virtual users*: pengguna virtual, *response time*: lamanya waktu antara request dan response dari server, *throughput*: penanganan request pada server, dan *error rate*: presentase error pada saat penanganan request berlangsung.

C. Studi Pustaka

Berdasarkan penelitian yang telah dilakukan, dapat dikatakan bahwa *database RDBMS* dan *NoSQL* adalah jenis *database* yang banyak digunakan dalam berbagai proses pengembangan aplikasi, dengan masing-masing keunggulan dan kelemahan. Untuk melakukan pengolahan data, diperlukan media penyimpanan yang baik. Tentu saja, penggunaan *Database Management System (DBMS)* sebagai mengandalikan pembuatan, pengolahan, dan pemeliharaan data yang efektif dan efisien adalah hal yang mendukung. Kemudian muncul masalah baru tentang jenis basis data mana yang lebih efisien dan responsif dalam proses transaksi data. Karena kebutuhan data akan terus meningkat di masa depan, diperlukan solusi untuk mempercepat waktu proses transaksi data antara kedua jenis basis data.

D. Kebutuhan Hardware dan Software

Kebutuhan melakukan indentifikasi *hardware* serta *software* yang dibutuhkan dalam proses pengukuran waktu respon pada *restapi*. Spesifikasi *hardware* dan *software* digunakan pada tabel I dan tabel II.

TABEL I
KETERANGAN HARDWARE

No	Item	Keterangan
1	*Processor*	Intel i7-10750H
2	*Memory Ram*	16 GB DDR4 2933 MHz
3	*Graphic Card Integrated*	Intel UHD Graphics
4	*Graphic Card External*	GTX 1660 TI
5	*Storage*	M.2 PCIe NVMe Samsung Evo 512 GB

TABEL III
KETERANGAN SOFTWARE

No	Item	Keterangan
1	*Windows*	11 / 64 Bit
2	*MySQL Xampp*	8.0.2
3	*Visual Studio Code*	1.80
4	*Node JS*	16.17.0
5	*Mongo DB*	6.0.7
6	*Mongo DB Compass*	1.38.2
7	*Postman*	10.15

E. Pengumpulan Data

Teknik pengumpulan data yang digunakan pada penelitian ini adalah dengan mengambil salah satu *database* buatan *Microsoft* yang bernama *Northwind database samples*. *Database Northwind* dibuat sebagai contoh untuk membantu pengembang dan profesional *IT* dalam memahami cara mendesain dan bekerja dengan basis data menggunakan produk *Microsoft* seperti *Microsoft Access*, *Microsoft SQL Server*, dan produk *database* lainnya. *Database Northwind* dirancang untuk mencakup skenario bisnis yang umum di industri ritel dan distribusi. Oleh karena itu, *database* ini menyimpan data terkait pelanggan, pesanan, produk, karyawan, dan logistik. *Database Northwind* memiliki struktur relasional dan terdiri dari beberapa tabel yang saling terkait. Tabel-tabel ini mencakup informasi seperti pelanggan, pesanan, produk, kategori produk, karyawan, dan lainnya. Berikut adalah contoh salah satu *table* pada *Northwind sample database*:

TABEL IIIII
CONTOH DATABASE NORTHWIND TABEL ORDERS

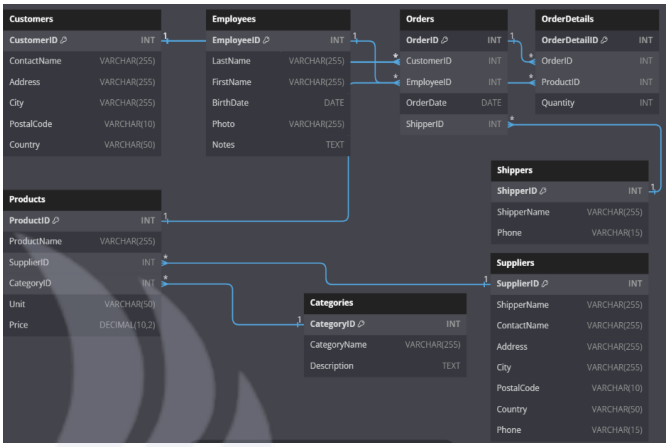
OrderID	*CustomerID*	*EmployeeID*	*OrderDate*	*ShipperID*
10248	90	5	1996-07-04 00:00:00	3
10249	81	6	1996-07-04 00:00:00	1
10250	34	4	1996-07-04 00:00:00	2
10251	84	3	1996-07-04 00:00:00	1
10252	76	4	1996-07-04 00:00:00	2

F. Perancangan Aplikasi yang Akan Diuji

Pada tahapan ini akan ceritakan langkah-langkah pada pengujian. Beberapa hal yang harus disiapkan sebelum melakukan proses pengukuran *response time* mengunakan *rest-api nodejs.*

1. Pembuatan Model Databse

   Dalam melakukan pembuatan model *database* yang akan diuji, penulis memakai dua *database* yaitu *MySQL* dan *MongoDB*. Sesuai dengan data yang akan diuji, maka diperoleh diagram *ERD* seperti berikut:



Gbr. 2 Diagram *ERD database Northwind*.

Pada schema *northwind sample database* terdapat delapan tabel yang terdiri dari tabel *Customers, Products, Employees, Orders, Shippers, OrderDetails, Categories, dan Suppliers.*

2. Pembuatan *Rest API*

   Dalam melakukan pembuatan *rest-api* untuk pengujian *database*, bahasa yang digunakan untuk proses *developement* adalah *JavaScript* menggunakan *framework ExpressJS* dan *runtime environtment NodeJS*. Dalam satu *service rest-api*, akan terdapat dua *database* yang akan diuji dan akan dibedakan berdasarkan *endpoint*. Berikut merupakan *endpoint* yang akan digunakan dalam pengujian *load testing* pada kedua *database*:

TABEL IV
SKENARIO ENDPOINT PADA RESTAPI NORTHWIND

Endpoint	*Operation*	*Table in operation*	*Record data*
create_shipper	CREATE	Shippers	5.000, 10.000,15.000
sum_avg	SUM, AVG	Customers, Orders, Orderdetails , Products	5.000, 10.000,15.000
count	COUNT	Customers , Orders, Employee	5.000, 10.000,15.000
and_not_exist	AND, NOT EXIST	Customers, Orders, Oerderdetails, Products	5.000, 10.000,15.000

or_in_exist	OR, IN, EXIST	Customers, Orders	5.000, 10.000,15.000
not_in	NOT IN	Customers, Orders, Orderdetails , Products	5.000, 10.000,15.000
update_shipper	UPDATE	Shippers	5.000, 10.000,15.000
delete_shipper	DELETE	Shippers	5.000, 10.000,15.000

Berdasarkan tabel IV menunjukan *endpoint* yang akan dibuat sebagai skenario pengujian *load test* pada masing-masing *database*. Berikut merupakan penjelasan singkat mengenai *endpoint* yang akan dibuat:

a) Endpoint Create

Endpoint ini berisi perintah unutk menyisipkan (*INSERT*) data baru ke dalam tabel *Shippers*. Endpoint ini menggunakan nilai yang diterima dari badan permintaan (*req.body*) untuk *ShipperName* dan *Phone*.

b) Endpoint Sum avg

Endpoint ini mencoba menemukan pelanggan (*Customers*) yang memiliki nilai total pesanan rata-rata di atas nilai total pesanan tertentu dari tabel *Orders* dan *OrderDetails*.

c) Endpoint Count

Endpoint ini memberikan informasi tentang jumlah pesanan yang dilakukan oleh pelanggan dari *'USA*. Serta mengelompokkan hasil berdasarkan kolom pelanggan.

d) Endpoint And not exist

Endpoint ini memberikan informasi terkait pelanggan (*Customers*) dari kota tertentu (*City*) yang telah melakukan pesanan (*Orders*), tetapi belum pernah melakukan pembelian produk tertentu (*Products*).

e) Endpoint In or exist

Endpoint ini berisi perintah untuk mengambil data pelanggan dari yang berada di *'USA'* atau memiliki pesanan dengan *ShipperID* '1' atau memiliki pesanan yang dibuat pada atau setelah tanggal '2023-01-01'.

f) Endpoint Not in

Endpoint ini berisi perintah untuk menemukan pelanggan (*Customers*) yang tidak pernah melakukan pembelian produk tertentu (*Products*) dengan kriteria tertentu dan yang berasal dari wilayah tertentu (*Region*).

g) Endpoint Update

Endpoint ini berfungsi untuk memperbarui nama perusahaan (*CompanyName*) menjadi *'Express Delivery Inc*.' dan nomor telepon (*Phone*) menjadi '(555) 123-4567' untuk *shipper* dengan *ShipperID* 4.

h) Endpoint Delete

Endpoint ini berfungsi untuk menghapus data *shipper* dengan *ShipperID* 4 dari tabel Shippers. Pastikan bahwa *ShipperID* yang di gunakan dalam klausa *WHERE* sesuai dengan data yang ingin di hapus.

G. Testing Aplikasi Menggunakan Postman

Setelah aplikasi sudah selesai dikembangkan, langkah selanjutnya adalah melakukan pengujian *load testing* pada *endpoint RestAPI northwind* dengan bantuan *tools Postman*. Pada proses pengujiannya menggunakan beberapa parameter seperti *virtual user*, *response time, troughput*, dan *error rate*. Dilansir dari halaman dokumentasi resmi *Postman* terkait dengan *perfomance test RestAPI* (*https://learning.postman.com/docs/collections/testing-api-performance/*), berikut adalah parameter indikator yang diukur ketika melakukan *load testing* menggunakan *Postman*:

1. *Virtual Users*

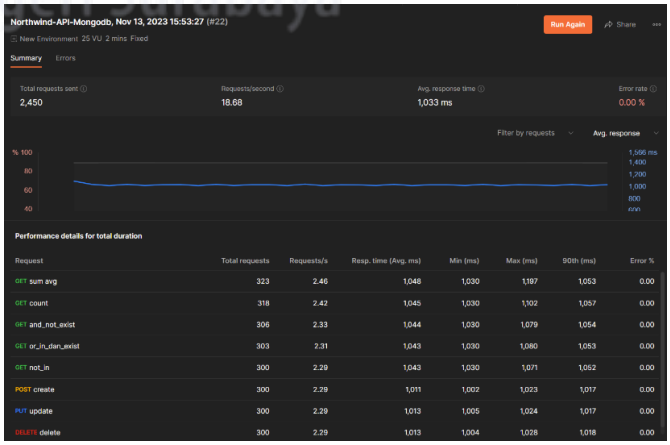
   Setiap pengguna virtual menyimulasikan perilaku pengguna di dunia nyata dengan menjalankan koleksi secara berulang.
2. *Response time*

   Indikator lamanya waktu antara *request* dan *response* dari server
3. *Troughput*

   Jumlah permintaan yang dikirim setiap detik selama pengujian kinerja, ukuran *throughput.*
4. *Error rate*

   Persentase permintaan yang menghasilkan kesalahan. Tanggapan selain tanggapan 2xx dianggap kesalahan.

Selanjutnya *Postman* akan menampilan keseluruhan output dan hasil akhir dari *load testing endpoint RestAPI* yang diuji. Berikut merupakan contoh *output load testing* menggunakan *Postman*:



Gbr. 3 Contoh *output* dari *Postman*

Berdasarkan gambar 3 diperoleh hasil di dari pengujian *load testing* pada *endpoint RestAPI*. Hasil pengujian tersebut menampilkan perolehan nilai dari masing-masing parameter pengujian meliputi *virtual numbers, response time, throughput*, dan *error rate*. Pengujian tersebut nantinya akan dilakukan 6 (enam) kali untuk setiap *endpoint*. Selanjutnya hasil dari pengujian masing-masing *endpoint database* akan disajikan dalam bentuk tabel seperti berikut:

TABEL V
TABEL HASIL PENGUJIAN MASING-MASING ENDPOINT KEDUA DATABASE

VU	***Response Time***	***Throughput***	***Error rate***	***Record data***
25				5.000
50				10.000
75				15.000

Dari hasil *load test endpoint database* tersebut nantinya akan diperoleh hasil dari 3 aspek yang akan diuji yaitu *response time, throughput*, dan *error rate*. Setelah memperoleh hasil *load test* tersebut selanjutnya akan dilakukan uji beda hasil menggunakan metode *independent T test* menggunakan *software SPSS* untuk mengetahui adanya perbedaan performa antara kedua *database*.

H. Analisis Hasil Pengujian

Setelah semua hasil *load testing* masing-masing *endpoint* kedua *database* tersebut sudah diperoleh. Kemudian penulis akan melakukan uji beda *Independent sample t-test* pada 3 parameter pengujian yaitu, *response time, throughput, error rate*. Seperti gambar berikut:

Group Statistics

	Jenis Database	N	Mean	Std. Deviation	Std. Error Mean
Response Time Database	Database MySQL	42	7.1905	2.83048	.43675
	Database MongoDB	42	34.0714	64.10182	9.89113

Independent Samples Test

		Levene's Test for Equality of Variances				
		F	Sig.	t	df	Sig. (2-tailed)
Response Time Database	Equal variances assumed	13.197	.000	-2.715	82	.008
	Equal variances not assumed			-2.715	41.160	.010

Gbr. 4 Contoh *Output Independent T Test*

Uji t independen sampel bertujuan untuk membandingkan rata-rata dua grup yang tidak saling berpasangan atau tidak saling berkaitan. *Rumus Polled Varians* digunakan untuk uji t untuk varian yang sama.:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

Gbr. 5 Rumus uji *t equal variance*

Hipotesis yang disusun adalah hipotesis dua arah dari 3 parameter pengujian, yaitu:

1. *Response time*
   H0 = Tidak terdapat perbedaan performa *response time* antara *MySQL* dan *NoSQL* pada *RestAPI NodeJS*.
   H1 = Terdapat perbedaan performa *response time* antara *MySQL* dan *NoSQL* pada *RestAPI NodeJS*.
2. *Throughput*
   H0 = Tidak terdapat perbedaan performa *throughput* antara *MySQL* dan *NoSQL* pada *RestAPI NodeJS*.
   H1 = Terdapat perbedaan performa *throughput* antara *MySQL* dan *NoSQL* pada *RestAPI NodeJS*.
3. *Error rate*
   H0 = Tidak terdapat perbedaan performa *error rate* antara *MySQL* dan *NoSQL* pada *RestAPI NodeJS*.
   H1 = Terdapat perbedaan performa *error rate* antara *MySQL* dan *NoSQL* pada *RestAPI NodeJS*.

Setelah dilakukan uji beda hasil dari masing-masing *endpoint*. Hasil akhir *load test* kedua *database* akan ditampilkan kedalam tabel hasil akhir dan dilakukan pengkategorian.

TABEL VI
PENGKATEGORIAN HASIL AKHIR PENGUJIAN LOAD TEST

Aspek	**Kategori**
Response Time	Cepat (Hijau)
	Sama (Kuning)
	Lambat (Merah)
Throughput	Banyak (Hijau)
	Sama (Kuning)
	Sedikit (Merah)
Error Rate	Sedikit (Hijau)
	Sama (Kuning)
	Banyak (Merah)

Berikut merupakan tabel hasil akhir pengujian *load test* dari *database MySQL* dan *database MongoDB*:

TABEL VII
HASIL AKHIR PENGUJIAN LOAD TEST KEDUA DATABASE

Endpoint	Aspek	*MySQL*	*MongoDB*
Create	***Response Time***		
	Throughput		
	Error Rate		
Sum avg	***Response Time***		
	Throughput		
	Error Rate		
Count	***Response Time***		
	Throughput		
	Error Rate		
And not exist	***Response Time***		
	Throughput		
	Error Rate		
Or in exist	***Response Time***		
	Throughput		
	Error Rate		
Not in	***Response Time***		
	Throughput		
	Error Rate		
Update	***Response Time***		
	Throughput		
	Error Rate		
Delete	***Response Time***		
	Throughput		
	Error Rate		

Tabel hasil akhir berisi kesimpulan akhir performa *load test* dari kedua *database* bedasarkan hasil dari pengujian *load test* dan uji beda hasil masing-masing *endpoint* kedua *database*.

III. *HASIL PENELITIAN DAN PEMBAHASAN*

Berikut merupakan tabel hasil akhir pengujian *load test* dari masing–masing *database*:

TABEL VIII
HASIL AKHIR KESELURUHAN LOAD TESTING KEDUA DATABASE

Endpoint	Aspek	*MySQL*	*MongoDB*
Create	***Response Time***	676.33	1338.78
	Throughput	6.21	1.42
	Error Rate	0	0.73
Sum avg	***Response Time***	2034.33	3611.39
	Throughput	5.14	1.61
	Error Rate	0	1.02
Count	***Response Time***	2016	3559.44
	Throughput	5.43	1.68
	Error Rate	0	3.68
And not exist	***Response Time***	1094.17	3030.72
	Throughput	5.81	1.68
	Error Rate	0	1.76
Or in exist	***Response Time***	1053.72	2813
	Throughput	5.2	1.42
	Error Rate	0	3.28
Not in	***Response Time***	1303.44	2651.67
	Throughput	6.27	1.54
	Error Rate	0	1.94
Update	***Response Time***	88.44	1134.28
	Throughput	6.71	1.42
	Error Rate	0	0.92
Delete	***Response Time***	33.83	1252.61
	Throughput	6.7	1.31
	Error Rate	0	0.8

Berikut merupakan pembahsan mengenai pengujian *load test* dan uji beda untuk setiap *endpoint* :

A. *Endpoint Create*

Pada pengujian *load test* dan uji beda untuk *endpoint create*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk aspek *response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 676.33ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 6.21 *request per seconds*. Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint create*, hal ini terjadi dikarenakan adanya perbedaan model data. Perbedaan dalam model data ini dapat berdampak pada kinerja, terutama dalam hal kecepatan pencarian dan penyimpanan data.

B. *Endpoint Sum avg*

Pada pengujian *load test* dan uji beda untuk *endpoint sum avg*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk aspek *response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 2034.33ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 5.14*request per seconds*. Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint sum avg*, hal ini terjadi dikarenakan pengaruh oleh *query* kompleks dan *join* 4 tabel. *MySQL* mungkin lebih cocok untuk aplikasi dengan

query kompleks dan hubungan yang kuat antar data, sementara *MongoDB* mungkin lebih unggul dalam situasi di mana skema data lebih dinamis.

C. Endpoint Count

Pada pengujian *load test* dan uji beda untuk *endpoint count*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk *aspek response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 2016ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 5.43*request per seconds*. Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint count*, hal ini terjadi dikarenakan pengaruh oleh *query* kompleks dan *join* 3 tabel. MySQL mungkin lebih cocok untuk aplikasi dengan *query* kompleks dan hubungan yang kuat antar data, sementara *MongoDB* mungkin lebih unggul dalam situasi di mana skema data lebih dinamis.

D. Endpoint And not exist

Pada pengujian *load test* dan uji beda untuk *endpoint* and *not exist*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk aspek *response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 1094.17ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 5.81*request per seconds.* Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint and not exist*, hal ini terjadi dikarenakan pengaruh oleh *query* kompleks dan *join* 4 tabel. *MySQL* mungkin lebih cocok untuk aplikasi dengan *query* kompleks dan hubungan yang kuat antar data, sementara *MongoDB* mungkin lebih unggul dalam situasi di mana skema data lebih dinamis.

E. Endpoint Or in exist

Pada pengujian *load test* dan uji beda untuk *endpoint or in exist*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk aspek *response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 1053.72ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 5.2*request per seconds.* Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint or in exist*, hal ini terjadi dikarenakan pengaruh oleh *query* kompleks dan *join* 2 tabel. *MySQL* mungkin lebih cocok untuk aplikasi dengan *query* kompleks dan hubungan yang kuat antar data, sementara *MongoDB* mungkin lebih unggul dalam situasi di mana skema data lebih dinamis.

F. Endpoint Not in

Pada pengujian *load test* dan uji beda untuk *endpoint not in*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk aspek *response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 1303.44ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 6.27*request per seconds*. Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint not in*, hal ini terjadi dikarenakan pengaruh oleh *query* kompleks dan *join* 4 tabel. *MySQL* mungkin lebih cocok untuk aplikasi dengan *query* kompleks dan hubungan yang kuat antar data, sementara *MongoDB* mungkin lebih unggul dalam situasi di mana skema data lebih dinamis.

G. Enpoint update

Pada pengujian *load test* dan uji beda untuk *endpoint update*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk aspek *response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 88.44ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 6.71*request per seconds*. Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki

performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint update*, hal ini terjadi dikarenakan adanya perbedaan model data. Perbedaan dalam model data ini dapat berdampak pada kinerja, terutama dalam hal kecepatan pencarian dan penyimpanan data.

H. Endpoint Delete

Pada pengujian *load test* dan uji beda untuk *endpoint delete*, dapat disimpulkan adanya perbedaan performa dari kedua *database*. Untuk aspek *response time database MySQL* mendapatkan hasil performa yang lebih baik dari *database MongoDB* yaitu sebesar 33.83ms. Lalu untuk aspek *throughput database MySQL* mendapatkan hasil performa yang lebih tinggi dari *database MongoDB* yaitu sebesar 6.7*request per seconds*. Selanjutnya untuk aspek *error rate database MySQL* juga mendapatkan hasil performa yang lebih baik dibandingkan dengan *database MongoDB* yaitu sebesar 0%. Dari hasil 3 (tiga) aspek tersebut dapat disimpulkan bahwa *database MySQL* memiliki performa yang lebih unggul dibandingkan dengan *database MongoDB* pada pengujian *load test endpoint delete*, hal ini terjadi dikarenakan adanya perbedaan model data. Perbedaan dalam model data ini dapat berdampak pada kinerja, terutama dalam hal kecepatan pencarian dan penyimpanan data.

Berdasarkan penelitian yang telah dilakukan, didapatkan hasil bahwa performa *database MySQL* lebih unggul dari pada *database MongoDB* dari 3 (tiga) aspek *response time*, *throughput*, dan *error rate* pada semua *endpoint*. Hal ini dikarenakan struktur *NorthwindDB* dirancang untuk aplikasi pemrosesan transaksi *online* (*online transaction processing, OLTP*). Oleh karena itu, membuat *database MySQL* memiliki performa lebih unggul dibandingkan dengan *database MongoDB*. Karena format penyimpanan data dalam *MySQL* membuatnya cocok untuk penggudangan data dan pemrosesan analitis *online*. Hal ini sesuai dengan *ACID (Atomicity, Consistency, Isolation, and Durability)*, yang berarti transaksi bersifat atomik, konsisten, terisolasi, dan stabil. Dengan ini, *database MySQL* berguna ketika bekerja dengan transaksi yang kompleks, seperti dalam kasus *e-commerce*, transaksi, dan keuangan. Berbeda dengan *database MongoDB* yang merupakan *database NoSQL*. Dimana *database MongoDB* itu lebih tepat jika bekerja dengan data tidak terstruktur dalam kasus penggunaan, seperti jejaring sosial, media, atau *IOT*. Karena *database MongoDB* tidak memiliki skema, *database MongoDB* merupakan pilihan yang tepat untuk menangani data yang terus berubah dan berkembang.

IV. PENUTUP

A. Kesimpulan

Penelitian ini dilakukan dengan tujuan agar dapat mengetahui perbandingan performa *load testing* antara *database MySQL* dan *database MongoDB* pada *Rest API NodeJS* studi kasus *NorthwindDB*. Parameter yang dihitung diambil dari model pengujian *load test* dengan bantuan *Postman* sehingga dapat diambil kesimpulan sebagai berikut :

1. Proses penganalisaan dimulai dengan membuat 2 macam *services* dengan *database* yang berbeda pada 1 *Rest API* yang sama menggunakan bahasa pemrogaman *javascript* dan *framework expressjs* dan menggunakan data dari *NorthwindDB*, kedua *services* dibuat dengan *database MySQL* dan *database MongoDB*. Selanjutnya akan dilakukan pengujian performa *load test* menggunakan *tools Postman* dengan 4 aspek yaitu *virtual users (VU), response time, throughput*, dan *error rate*.
2. Hasil penelitian ini menunjukkan bahwa *database MySQL* lebih unggul karena memiliki nilai yang lebih baik dalam 3 (tiga) aspek yaitu *response time, throughput*, dan *error rate* dibandingkan dengan *database MongoDB*.

B. Saran

Untuk studi kasus ini, *database MySQL* memiliki performa *load testing* yang lebih baik dibandingkan dengan *database MongoDB*, tetapi ini tidak berarti bahwa *database MongoDB* tidak dapat dipertimbangkan ketika ingin mengembangkan aplikasi untuk masa depan karena setiap *database* memiliki kegunaan dan kegunaannya sendiri.. Dalam penelitian ini, *database MySQL* memiliki performa *load testing* yang lebih baik karena struktur *NorthwindDB* dirancang khusus untuk *database MySQL*. Diharapkan bahwa penelitian lanjutan akan dapat menganalisis perbandingan elemen seperti skalabilitas, fleksibilitas, keamanan, efisiensi biaya, dan efisiensi waktu pengembangan. karena penelitian dari sisi lain dapat membantu pengembang aplikasi memilih *database* yang tepat untuk studi kasus aplikasi.

REFERENSI

[1] Padillah, R. (2021). Implementasi Revolusi Industri (4.0) Pada Ukm Ayam Broiler Melalui Mesin Pakan Ayam Otomatis Berbasis Internet Of Things (IoT). JATI EMAS (Jurnal Aplikasi Teknik Dan Pengabdian Masyarakat), 5(1), 1–4. https://doi.org/10.36339/JE.V5I1.382.

[2] Gunawan, R., Aulia, S., Supeno, H., Wijanarko, A., Uwiringiyimana, J. P., Mahayana, D., & Teknik, S. (2021). Adiksi Media Sosial dan Gadget bagi Pengguna Internet di Indonesia. TECHNO-SOCIO EKONOMIKA, 14(1), 1–14. https://doi.org/10.32897/TECHNO.2021.14.1.544.

[3] Silalahi, M., & Wahyudi, D. (2018). Computer Based Information System Journal PERBANDINGAN PERFORMANSI DATABASE MONGODB DAN MYSQL DALAM APLIKASI FILE MULTIMEDIA BERBASIS WEB. CBIS JOURNAL, 06(01). http://ejournal.upbatam.ac.id/index.php/cbis\http://ejournal.upbatam.ac.id/index.php/cbisM.

[4] Rahmatullah, S., & Ndaru, M. (2023). SISTEM INFORMASI PENJUALAN SEMBAKO PADA TOKO BAPAK NASRUL BERBASIS WEB. Jurnal Informatika Software Dan Network), 04(01), 1–6.

[5] Fatman, Y., Khoirun Nafisah, N., & Bendoro Jembar Pambudi, P. (2023). Implementasi Payment Gateway dengan Menggunakan Midtrans pada Website UMKM Geberco. Jurnal KomtekInfo, 64–72. https://doi.org/10.35134/komtekinfo.v10i2.364.

[6] Raharjo, M., Napiah, M., & Anwar, R. S. (1045). Perancangan Sistem Informasi Dengan PHP Dan MYSQL Untuk Pendaftaran Sekolah Di Masa Pandemi (Issue 18). http://jurnal.bsi.ac.id/index.php/co-science.

[7] Hadi Winata, R., & Yunanto Putra, Y. (2021). ANALISIS PERBANDINGAN PERFORMA WAKTU RESPONS KUERI ANTARA MySQL PHP 7.4.2, PostgreSQL, dan MongoDB COMPARISONAL ANALYSIS OF PERFORMANCE QUESTION RESPONSE TIME BETWEEN MySQL PHP 7.4.2, PostgreSQL, and MongoDB (Vol. 38).

[8] Fakultas Matematika dan Ilmu Pengetahuan Alam, J. (2020). Analisis Kinerja Web Server pada SIM Manajemen Diklat Poltekpel Sorong Menggunakan RDBMS MySQL dan MariaDB. In Journal of System and Computer Engineering (JSCE) ISSN (Vol. 1, Issue 1).

[9] Fadli, A., Zulfa, M. I., Widhi Nugraha, A. W., Taryana, A., & Aliim, M. S. (2020). Analisis Perbandingan Unjuk Kerja Database SQL dan Database NoSQL Untuk Mendukung Era Big Data. JURNAL NASIONAL TEKNIK ELEKTRO, 9(3). https://doi.org/10.25077/jnte.v9n3.774.2020

[10] Sari, A. S., & Hidayat, R. (2022). Designing website vaccine booking system using golang programming language and framework react JS. Journal of Information System, Informatics and Computing Issue Period, 6(1), 22–39. https://doi.org/10.52362/jisicom.v6i1.760