Procedia of Engineering and Life Science Vol. 2. No. 2 June 2022
Seminar Nasional & Call Paper Fakultas Sains dan Teknologi (SENASAINS 4th)
Universitas Muhammadiyah Sidoarjo

Query Response Time Comparison SQL and No SQL for Contact Tracing Application

Perbandingan Waktu Respons Kueri SQL dan Tanpa SQL untuk Aplikasi Pelacakan Kontak

Arik Bagus Setyawan, Irwan Alnarus Kautsar, Nuril Lutvi Azizah
{setyawanarik@gmail.com, irwan@umsida.ac.id, nurillutviazizah@umsida.ac.id}

Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Muhammadiyah Sidoarjo

Abstract. *During the current Covid-19 pandemic, it is necessary to have facilities to track community activities to make it easier to cope with the Covid-19 pandemic situation which is still a threat to the community. Contact tracing application (Contact Tracing) is one solution that can be used in this regard. By utilizing Contact Tracing, people's activities can be recorded when traveling to meet other people so that they can generate a history that can be used to track someone if one of them is infected with Covid-19. With this application, data storage (database) is very important because the application will generate and consume a lot of data. In this study, the researcher attempted to compare the query speed of the common data storage (database) that is usually used, namely Relational Database (MySQL) with Non-Relational Database (MongoDB). The results of the experiments conducted by the researchers resulted in the Non-Relational Database (MongoDB) suitable for use when the data has reached large amounts because the query speed is quite fast and stable. Meanwhile, Relational Database performance is still reliable when the data query needsare only on a small scale.*

Keywords - *Database Comparison; MongoDB; Non-Relational Database; Relational Database*

Abstrak. *Pada masa pandemi Covid-19 sekarang ini, perlu adanya fasilitas untuk melakukan pelacakan aktivitas masyarakat guna mempermudah untuk menanggulangi situasi pandemi Covid-19 yang masih menjadi ancaman bagi masyarakat. Aplikasi telusur kontak (Contact Tracing) merupakan salah satu solusi yang dapat digunakan dalam hal tersebut. Dengan memanfaatkan Contact Tracing, aktivitas masyarakat dapat direkam ketika bepergian untuk bertemu orang lain sehingga dapat menghasilkan history yang bisa digunakan untuk melacak seseorang jika salah satu dari mereka terjangkit Covid-19. Dengan adanya aplikasi tersebut, tempat penyimpanan data (Database) merupakan hal yang sangat penting karena aplikasi tersebut nantinya akan menghasilkan serta mengkonsumsi data yang sangat banyak. Pada penelitian ini, peneliti berupaya untuk membandingkan kecepatan query tempat penyimpanan data (Database) umum yang biasanya digunakan yaitu Relational Database (MySQL) dengan Non-Relational Database (MongoDB). Hasil uji coba yang dilakukan peneliti menghasilkan Non-Relational Database (MongoDB) cocok digunakan ketika data sudah mencapai jumlah besar karena kecepatan query yang cukup cepat dan stabil. Sedangkan Relational Database performanya masih dapat diandalkan ketika kebutuhan query data hanya dalam skala kecil.*

Kata Kunci - *Database Comparison; MongoDB; Non-Relational Database; Relational Database*

I. PENDAHULUAN

Database dapat didefinisikan sebagai kumpulan data yang disimpan pada *hard drive* sistem komputer. Data yang disimpan dalam *database* biasanya diatur untuk memodelkan aspek-aspek yang mendukung proses yang memerlukan penyimpanan dan pengambilan informasi [1]. Saat ini ada dua macam *database*, yaitu *Relational Database* dan *Non- Relational Database* [2]. Saat ini *Relational Database* sangat popular serta umum digunakan untuk menyimpan data yang sudah terstruktur, contohnya Microsoft SQL Server, Oracle MySQL, PostgreSQL [3]. *Relational Database* banyak digunakan karena mempunyai sekumpulan fitur yang kaya, kecepatan *query* dan manajemen transaksi yang bagus. Saat ini *Non-Relational Database* juga mulai popular dan banyak digunakan. *Non-Relational Database* menawarkan performa skalabilitas serta manajemen data yang sangat fleksibel dibandingkan dengan *Database Relational* [4], contoh dari *Non-Relational Database* antara lain MongoDB, Cassandra, Neo4j. MongoDB merupakan sistem database berorientasi dokumen yang menyediakan berbagai tingkat konsistensi untuk memungkinkan aplikasi *client* memilih pertukaran data yang ingin mereka buat dalam hal konsistensi dan latensi saat dioperasikan [5]. *NoSQL* menyediakan model data yang lebih fleksibel, skalabilitas yang lebih tinggi, dan juga kinerja yang unggul sambal tetap menggabungkan beberapa fitur utama dari *Relational Database* [6]. Berbeda dengan *Non-Relational Database*, *Relational Database* menyimpan data dalam bentuk baris dan kolom dalam tabel dan harus menentukan skema sebelumnya berdasarkan kebutuhan untuk mengontrol relasi antar tabel[7] [8].

Penelitian ini dilakukan untuk mengetahui perbandingan performa *query* pada *Non-Relational Database* dengan

Relational Database. Penelitian ini dilakukan dengan menggunakan *query* sederhana terhadap data yang terstruktur. *Query* terdiri dari pengambilan data dari lebih dari satu tabel dengan menggunakan operasi gabungan.

II. METODE

Penelitian ini berfokus untuk mendapatkan hasil kecepatan *query* antara *Non-Relational Database* dan *Relational Database* dalam pengolahan data yang sudah terstruktur. Penelitian ini dilakukan dengan menggunakan beberapa langkah yang terdiri sebagai berikut:

- Melakukan pendekatan dengan menggunakan studi literatur tentang *Non-Relational Database* dan *Relational Database* guna memahami penyimpanan data serta model data *Non-Relational Database* dan *Relational Database*. Pada tahap ini pemahaman dan pembelajaran konsep tersebut dilakukan dengan membaca artikel ilmiah, buku, dan makalah yang ada di bidang tersebut.
- Analisa *query* dan model data pada *Non-Relational Database* dan *Relational Database*. Analisa ini menghasilkan model data untuk *database* serta *query* yang akan dieksekusi dalam percobaan yang akan dijalankan. Alasisa ini juga diperlukan untuk membangun program sebagai alat pengetesan antara dua *database* tersebut.
- Membangun aplikasi yang terdiri dari metode untuk mengeksekusi *query* serta mendukung percobaan yang akan dilakukan. Aplikasi yang akan dibangun akan menampilkan waktu *query* untuk setiap eksekusi yang akan dijalankan.
- Percobaan akan dilakukan dengan menggunakan aplikasi yang sudah dibangun untuk mengukur waktu yang dibutuhkan untuk mengeksekusi sebuah perintah *query*.
- Menyimpulkan hasil eksperimen *query database* yang sudah dilakukan untuk menghasilkan perbandingan antara *Non-Relational Database* dan *Relational Database*.

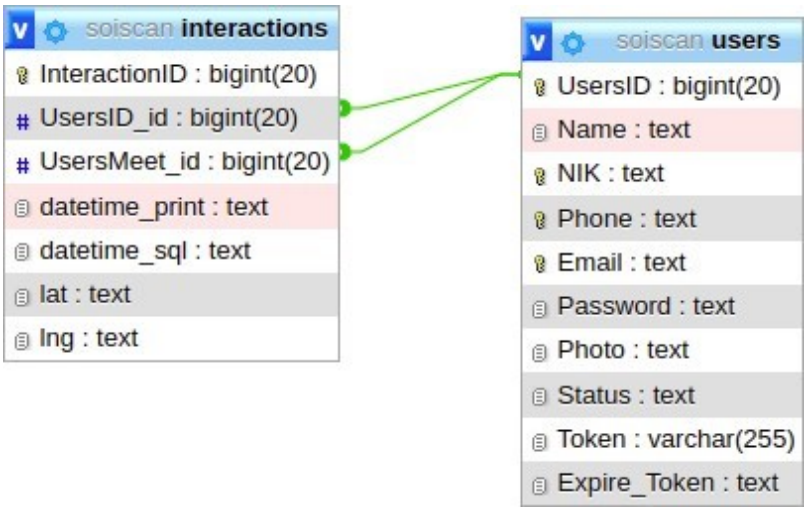
Untuk mengetahui perbandingan antara *Graph Base Database* dengan *Relational Database* maka di sini peneliti akan melakukan pengujian dengan beberapa dataset yang sudah di-*generate*. Terdapat tiga tahap pengujian yang sudah peneliti siapkan.

- *Query* mengambil semua data *user* dan interaksi yang saling berelasi.
  Pada tahap ini peneliti akan membandingkan kecepatan *query* dari *Graph Base Database* dan *Relational Database* menggunakan 10,100, dan 1000 data *user* yang melakukan 100, 1000, dan 10000 interaksi. Semua data *user* dan interaksi akan diambil tanpa pengecualian.
- *Query user* secara spesifik dan interaksi yang terkait.
  Pada tahap ini peneliti akan melakukan *query* secara spesifik untuk entitas *user* dengan menggunakan parameter email yang terdapat pada entitas *user*. Peneliti membandingkan kecepatan *query* dari *Graph Base Database* dan *Relational Database* menggunakan 10,100, dan 1000 data *user* yang melakukan 100, 1000, dan 10000 interaksi.
- *Query* interaksi berdasarkan rentang waktu tertentu.
  Tahap pengujian terakhir peneliti akan melakukan *query* secara spesifik terhadap rentang waktu interaksi. Uji coba kecepatan *query Graph Base Database* dan *Relational Database* menggunakan 10,100, dan 1000 data *user* yang melakukan 100, 1000, dan 10000 interaksi.

III. HASIL DAN PEMBAHASAN

A. Representasi data relational database

Data sampel yang akan digunakan terlebih dahulu akan di *generate* oleh program(*dummy*). *Database* telusur kontak akan mempunyai dua tabel yang saling berhubungan. Setiap tabel mempunyai primary key sebagai penanda identitas dan jika berhubungan dengan tabel lain, maka *primary key* akan menjadi foreign key pada tabel yang berelasi isi tabel dapat dilihat pada gambar 1.



Gambar 1. Representasi Data *Relational Database*

B. Representasi data non-relational database

Representasi data pada *Non-Relational Database* merupakan hasil analisa data yang dilakukan terhadap data yang ada pada *Relational Database* dengan mengubah skema yang ada. Analisa yang dijalankan meliputi mengubah tabel dan hubungan relasi antar tabel menjadi *JSON Object*. Skema representasi data pada *Non-Relational Database* dapat dilihat pada gambar 2.


```
{
	"id" : "6283c1fa63f5789196f6b377",
	"name" : "Leo Myklebost",
	"NIK" : "32595531086866",
	"Phone" : "22678457706",
	"Email" : "leo.myklebost@example.com",
	"Password" : "yaya",
	"photo" : "user_profile_default.png",
	"status" : "Negatif",
	"token" : "",
	"expire_token" : "",
	"interactions" : [
		{
			"user" : {
				"id" : "6283c1f963f5789196f6b36f",
				"name" : "Jenny Riley",
				"NIK" : "47674831027429",
				"Phone" : "39733818296",
				"Email" : "jenny.riley@example.com",
				"Password" : "sebastia",
				"photo" : "user_profile_default.png",
				"status" : "Negatif",
				"token" : "",
				"expire_token" : ""
			},
			"datetime_print" : "2020-11-15 20:14:13",
			"datetime_sql" : 1605446053.064718,
			"lat" : -7.428539,
			"lng" : 112.669798
		}
	]
}
```


Gambar 2. Representasi Data *Non-Relational Database*

C. Analisa query

Query akan dieksekusi guna mendapatkan hasil pengujian kecepatan *query* antara *Non-Relational Database* dan *Relational Database* untuk memproses *query* sederhana. Untuk memperkuat hasil pengujian *query* kecepatan *Relational Database* dan *Non-Relational Database*, analisa dilakukan untuk mencari representasi *query* terbaik. *Query* yang akan dijalankan meliputi:

1. *Query* mengambil semua data user dan interaksi yang saling berelasi.
2. *Query* user secara spesifik dan interaksi yang terkait.
3. *Query* interaksi berdasarkan rentang waktu tertentu.

D. Analisa pengembangan aplikasi

Aplikasi akan dibangun menggunakan bahasa pemrograman *Python*. Aplikasi ini dibangun bertujuan untuk men-*generate* data *dummy* menggunakan fungsi *random.uniform()* pada *library Python Numpy* untuk menghasilkan

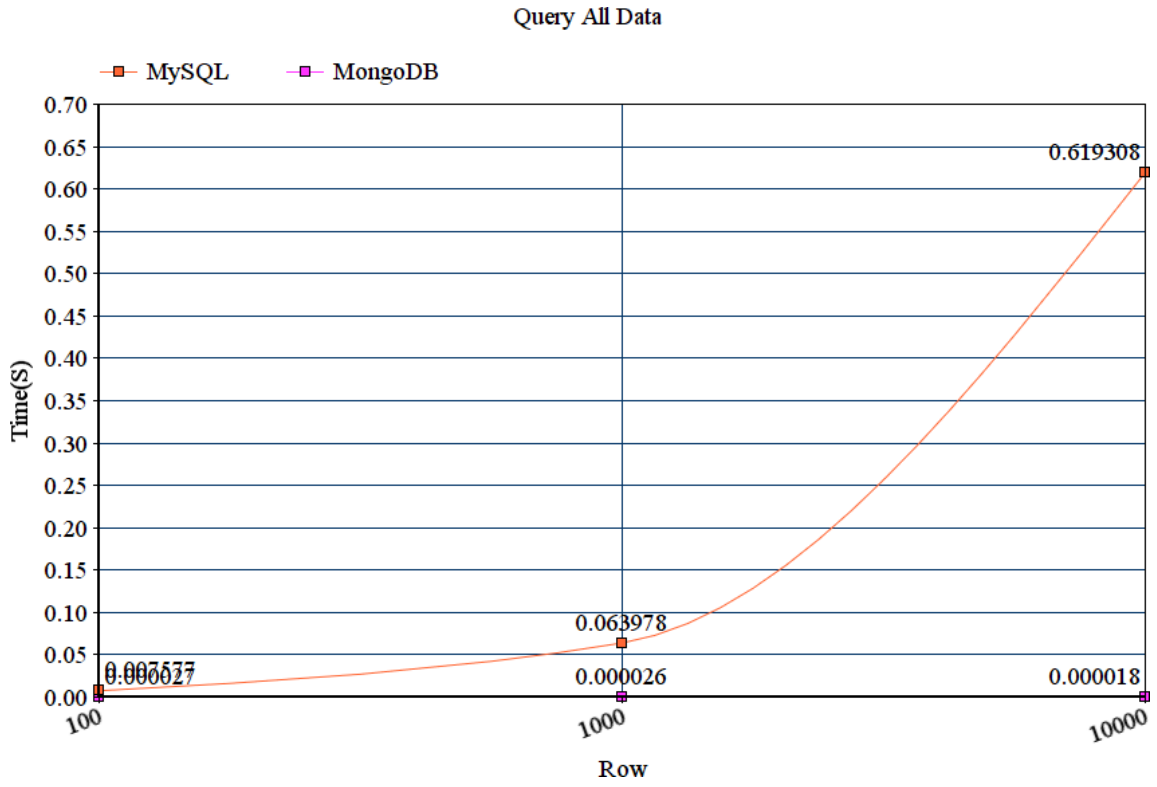
lokasi titik koordinat secara acak. *Library NumPy* merupakan struktur data yang secara efisien menyimpan dan mengakses *array* multidimensi [10]. Dengan menggunakan *library NumPy* kita dapat menghasilkan bilangan acak yang terdistribusi untuk keperluan dataset [11]. Data yang sudah dihasilkan akan digunakan untuk bahan pengujian performa antara *Relational Database* dan *Non-Relational Database.* Selain itu aplikasi ini dibangun untuk menyetarakan lingkungan eksperimen yang akan dilaksanakan sehingga hasil yang didapat tidak ada perilaku yang berbeda. Secara keseluruhan terdapat beberapa aspek perbandingan yang dihasilkan dari analisis yang dilakukan pada penelitian ini. Perbandingannya bisa dilihat pada Tabel 1.

Tabel 1. Perbandingan *Non-Relational Database* VS *Relational Database*

	Relational Database	*Non-Relational Database*
Struktur penyimpanan	Data disimpan didalam tabel yang terdiri dari baris dan kolom.	Data disimpan dalam bentuk JSON Dokumen.
Tipe Relasi	One-to-One, One-to-Many, Many-to-Many. Menggunakan primary key dan foreign key untuk membuat sebuah relasi.	Menggunakan metode denormalisasi data dan menyajikan semua data dalam suatu objek pada satu record
Query	Menggunakan bahasa SQL.	Menggunakan bahasa MQL.

E. Pengujian

Pada pengujian, MongoDB dan MySql diinstall pada mesin yang menjalankan CPU Intel Core i5 8250U @1.6GHz, RAM 8GB, dan sistem operasi Ubuntu 20.04LTS. Pengujian yang akan dilakukan pada penelitian ini memiliki tiga skenario pengujian. Skenario pertama akan memiliki 10 user dengan 100 interaksi, skenario kedua akan memiliki 100 user dengan 1000 interaksi, skenario ketiga akan memiliki 1000 user dengan 10000 interaksi. Setiap *query* akan dieksekusi oleh program sebanyak 10 kali dan akan diambil waktu rata-rata guna mendapatkan hasil akhir. Perbandingan hasil pengujian dapat dilihat pada grafik berikut.



Gambar 3. Pengujian Mengambil Semua Data User dan Interaksi

Gambar 3 memperlihatkan perbandingan waktu hasil *query* dari pengujian pertama. Terlihat pada *query* yang

menggunakan 100 data interaksi dan 10 data user, perbedaan kecepatan antara *Relational Database* dan *Non-Relational Database* terlihat signifikan, perbedaan kecepatan *query* antara dua *database* tersebut *Non-Relational Database MongoDB* lebih cepat 198.5% dibandingkan dengan *Relational Database*. Pada pengujian dengan menggunakan 1000 data interaksi dengan 100 data user, *Non-Relational Database MongoDB* masih lebih cepat 199% dibandingkan *Relational Database*. Perbedaan kecepatan *query* terbesar terdapat pada pengujian dengan menggunakan 10000 data interaksi dan 1000 data user. Pada pengujian tersebut *Non-Relational Database MongoDB* 199% lebih cepat dibandingkan dengan *Relational Database*. Pengujian tersebut menunjukkan *Non-Relational Database MongoDB* cenderung lebih cepat dan stabil dalam menangani *query* dengan jumlah data yang besar sekalipun. Sedangkan berbeda dengan *Non-Relational Database MongoDB*, *Relational Database* dapat menangani *query* dengan jumlah data kecil dan sedang dengan kecepatan yang tidak terlalu lambat. Akan tetapi *Relational Database* akan langsung terasa lambat jika dihadapkan dengan jumlah data yang besar.

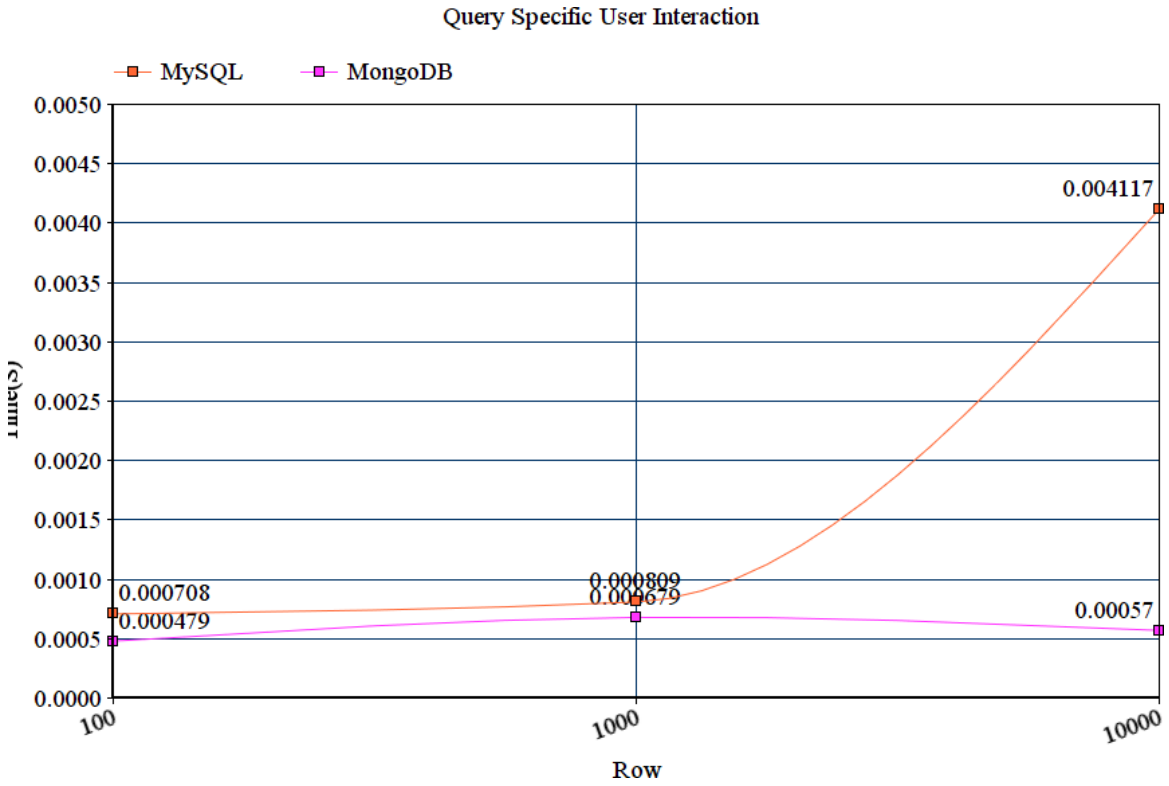
Contoh *Query* yang dieksekusi pada gambar 3 adalah:

- *Non-Relational Database*:

```
db.users.find()
```

- *Relational Database*:

```
SELECT * FROM users a LEFT JOIN interactions b ON a.UsersID=b.UsersID_id
RIGHT JOIN users c ON b.UsersMeet_id=c.UsersID;
```



Gambar 4. Pengujian Mengambil Data Interaksi Dengan User Secara Spesifik

Pada pengujian kedua seperti yang terlihat pada gambar 4 menunjukkan dimana pada skenario pencarian data, *Relational Database* dan *Non-Relational Database* kecepatan *query* masih sejajar. Pada pengujian dengan menggunakan 100 data interaksi dan 10 data user, *MongoDB* lebih cepat dibandingkan dengan *MySQL* sebesar 38.5%. Pada pengujian dengan menggunakan 1000 data interaksi dan 100 data user, *MongoDB* masih lebih cepat 7% dibandingkan dengan *MySQL*. Sedangkan pengujian dengan menggunakan 10000 data interaksi dan 1000 data user, terlihat bahwa *MongoDB* jauh lebih stabil dan cepat 151% dibandingkan dengan *Relational Database MySQL*. Hasil dari pengujian kedua ini menunjukkan bahwa performa *query Non-Relational Database MongoDB* sangat cepat dan stabil meski jumlah data yang dicari dalam jumlah besar. Performa *Relational Database* MySQL masih terbilang cukup cepat pada pencarian data dengan jumlah kecil dan sedang, akan tetapi ketika data yang dicari ada pada jumlah yang besar, performanya akan terasa lambat.

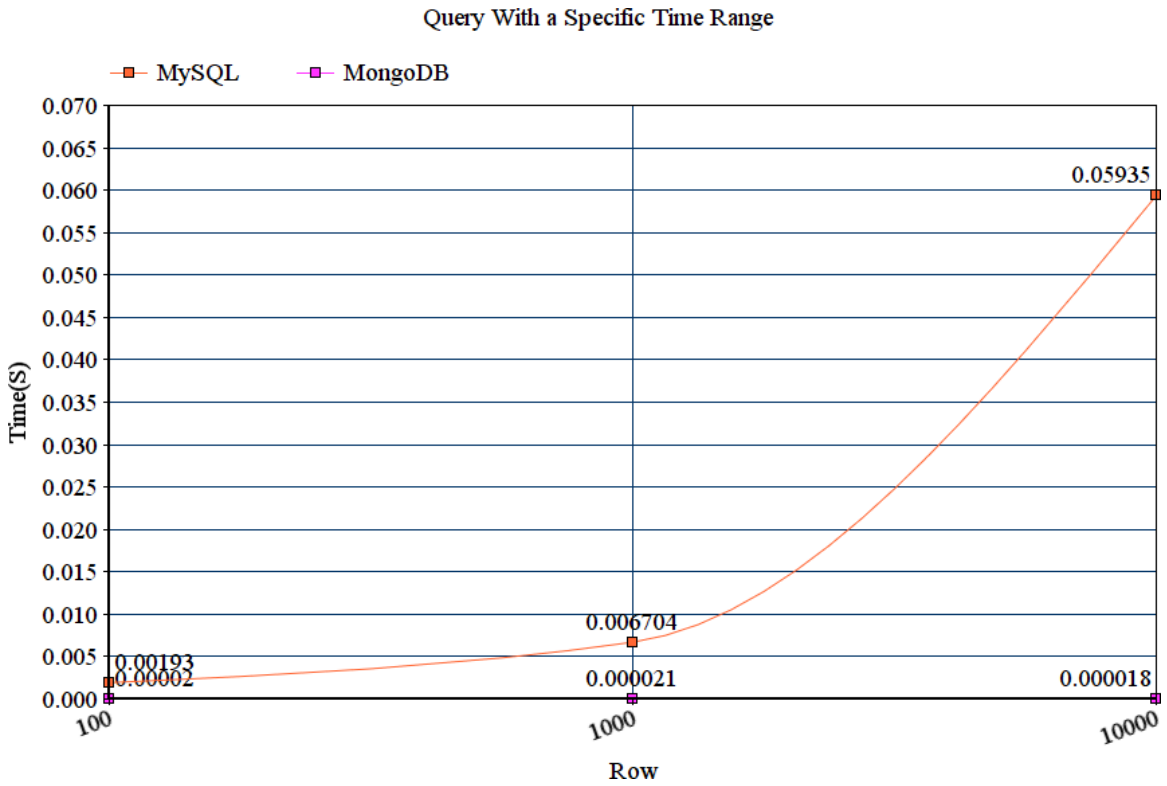
Contoh *Query* yang dieksekusi pada gambar 4 adalah:

- *Non-Relational Database*:

```
db.users.find_one({"Email":Email})
```

- *Relational Database*:

```
SELECT * FROM users a LEFT JOIN interactions b ON a.UsersID=b.UsersID_id
RIGHT JOIN users c ON b.UsersMeet_id=c.UsersID where a.Email =
random.user@example.com';
```



Gambar 5. Pengujian Mengambil Data Interaksi Dengan Rentang Waktu Tertentu

Gambar 5 memperlihatkan hasil dari pengujian ketiga yang telah dilaksanakan. Terlihat pada pengujian tersebut dengan menggunakan 100 data interaksi dan 10 data user, *MongoDB* lebih cepat 195% dibandingkan dengan *Relational Database*. Hal yang sama terlihat pada pengujian menggunakan 1000 data interaksi dan 100 data user, dimana *Non-Relational Database* lebih cepat 198% dibandingkan dengan *Relational Database*. Pengujian dengan menggunakan 10000 data interaksi dan 1000 data user juga menunjukkan hal yang serupa dimana *Non-Relational Database MongoDB* masih lebih cepat dibandingkan dengan *Relational Database MySQL* dengan selisih 199% lebih cepat. Pengujian terakhir ini menunjukkan kestabilan, dan kecepatan *Non-Relational Database* dalam melakukan *query* data dari jumlah kecil sampai besar. Terlihat performa *MongoDB* cukup stabil dan konsisten dibandingkan dengan *MySQL*. Seperti di pengujian kedua, pada pengujian menggunakan 100 data interaksi dan 10 data user *Relational Database* tidak terlalu tertinggal karena data yang di *query* hanya sedikit. Ketika data hasil *query* mulai banyak, performa *Relational Database* akan mulai lebih lambat. Perbedaan terbesar cukup terlihat ketika menjalankan *query* pada 10000 data interaksi dan 1000 data user, performa dari *Relational Database* sangat lambat dibandingkan dengan *Non-Relational Database* yang konsisten.

Contoh *query* yang dieksekusi pada gambar 5 adalah:

- *Non-Relational Database*:

```
db.users.find({"interactions":{"$elemMatch":{"datetime_sql":{"$gt":
1594490206.310136,"$lt":1597168606.310136}}}})
```

- *Relational Database*:

```
SELECT * FROM users a LEFT JOIN interactions b ON a.UsersID=b.UsersID_id
RIGHT JOIN users c ON b.UsersMeet_id=c.UsersID where b.datetime_sql between
1594490206.310136 and 1597168606.310136;
```

IV. KESIMPULAN

Pengujian pada penelitian ini memperlihatkan bahwa pemilihan *database* merupakan sebuah aspek yang perlu diperhitungkan ketika membangun sebuah aplikasi. *Relational Database* akan cocok dipasangkan dengan aplikasi yang kebutuhan penyimpanan datanya tidak terlalu besar. Performa *Non-Relational Database* akan terlihat cepat dan stabil meskipun jumlah data yang dikonsumsi dalam jumlah besar. Performa tersebut ditunjukkan dengan semua pengujian yang telah dilakukan. Dimana perbedaan kecepatan sangat terlihat. Namun pada pengujian kedua ketika data yang diambil dalam jumlah sedikit, *Relational Database* masih dapat diandalkan. Data yang ditampung oleh aplikasi telusur kontak lambat laun akan menjadi besar seiring bertambahnya waktu. Maka dari itu *Non-Relational Database* dapat dijadikan solusi ketika kebutuhan penyimpanan data yang besar nantinya.

UCAPAN TERIMA KASIH

Terimakasih disampaikan kepada Universitas Muhammadiyah Sidoarjo serta dosen pembimbing yang telah membantu dalam penelitian ini.

REFERENSI

[1] M. Malik and T. Patel, “Database Security - Attacks and Control Methods,” *Int. J. Inf. Sci. Tech.*, vol. 6, pp. 175–183, Mar. 2016, doi: 10.5121/ijist.2016.6218.

[2] F. A. Bhaswara, R. Sarno, and D. Sunaryono, “Perbandingan Kemampuan Database NoSQL dan SQL dalam Kasus ERP Retail,” *J. Tek. ITS*, vol. 6, no. 2, Art. no. 2, Sep. 2017, doi: 10.12962/j23373539.v6i2.24031.

[3] S. Suryadi, “Implementasi Normalisasi Dalam Perancangan Database Relational,” *U-NET J. Tek. Inform.*, vol. 3, no. 2, Art. no. 2, Aug. 2019, doi: 10.52332/u-net.v3i2.7.

[4] K. Fraczek and M. Plechawska-Wojcik, “Comparative Analysis of Relational and Non-relational Databases in the Context of Performance in Web Applications,” in *Beyond Databases, Architectures and Structures. Towards Efficient Solutions for Data Analysis and Knowledge Representation*, Cham, 2017, pp. 153–164. doi: 10.1007/978-3-319-58274-0_13.

[5] W. Schultz, T. Avitabile, and A. Cabral, “Tunable consistency in MongoDB,” Proc. VLDB Endow., vol. 12, no. 12, pp. 2071–2081, Aug. 2019, doi: 10.14778/3352063.3352125.

[6] M. M. Patil, A. Hanni, C. H. Tejeshwar, and P. Patil, “A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retrieval operations using a web/android application to explore load balancing — Sharding in MongoDB and its advantages,” in 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Feb. 2017, pp. 325–330. doi: 10.1109/I-SMAC.2017.8058365.

[7] D. Damodaran B, S. Salim, and S. M. Vargese, “Performance Evaluation of MySQL and MongoDB Databases,” *Int. J. Cybern. Inform.*, vol. 5, no. 2, pp. 387–394, Apr. 2016, doi: 10.5121/ijci.2016.5241.

[8] S. Rautmare and D. M. Bhalerao, “MySQL and NoSQL database comparison for IoT application,” in *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, Oct. 2016, pp. 235–238. doi: 10.1109/ICACA.2016.7887957.

[9] I. K. G. Sudiartha, I. N. E. Indrayana, I. W. Suasnawa, S. A. Asri, and P. W. Sunu, “Data Structure Comparison Between MySql Relational Database and Firebase Database NoSql on Mobile Based Tourist Tracking Application,” *J. Phys. Conf. Ser.*, vol. 1569, p. 032092, Jul. 2020, doi: 10.1088/1742-6596/1569/3/032092.

[10] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, Art. no. 7825, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[11] P. Charbonneau, “C. RANDOM NUMBERS AND WALKS,” in *C. RANDOM NUMBERS AND WALKS*, Princeton University Press, 2017, pp. 321–337. doi: 10.1515/9781400885497-016.