Jurnal Ilmiah Sinus (JIS) Vol : 21, No. 2, Juli 2023
ISSN (Print) : 1693-1173 , ISSN (Online): 2548-4028

Comparison of Measuring RDBMS Database Performance with NoSQL on Electronic Medical Record Systems (EMR)

Erwin Apriliyanto[1)]
[1)]Teknik Komputer, Universitas Muhammadiyah Karanganyar
[1)] itanalisterwin@gmail.com

ABSTRACT

An electronic medical record is a system for storing and managing patient medical data in electronic form. EMR replaces traditional medical record systems that still use paper, enabling doctors and other medical personnel to access patient medical records more quickly and easily from anywhere, as long as they are connected to the same network. Patient medical data is stored in electronic formats, such as text, images, and test results. Given that many application developers are starting to provide fast services with optimal performance in their applications, especially ERM, and that, over time, the use of larger data requires a variety of data complexity, both structured and unstructured, conventional database management systems have not yet responded effectively to this challenge. The purpose of this research is to test the speed of PostgreSQL and NoSQL with insert, select, and delete commands. Test Method Using 1,000 records, 5,000 records, and 10,000 records, with each record tested three times, the average is taken. With 1,000 data records, the NoSQL database is 471 times faster; with 5,000 data records, the NoSQL database is 355 times faster; and the select command for 1 NoSQL database table is 8 times faster than a PostgreSQL database, while for 1 database table, PostgreSQL above is faster.

Keywords: *NoSQL, MongoDB, PostgreSQL, ERM*

I. PENDAHULUAN

Sebelum adanya *EMR*, catatan medis pasien disimpan secara manual dalam format kertas, yang memakan waktu dan biaya yang signifikan untuk pengarsipan dan pemeliharaan. Selain itu, sulit untuk berbagi data antara penyedia layanan kesehatan dan rumah sakit, sehingga memperlambat proses pengambilan keputusan dan diagnosis. *EMR* memungkinkan data kesehatan pasien untuk disimpan dalam format elektronik, yang membuat akses data menjadi lebih mudah dan cepat.

Penerapan *EMR* juga membantu mengurangi risiko kesalahan dalam pengelolaan catatan medis pasien, termasuk kesalahan dalam pengisian formulir, penulisan resep, dan kesalahan dalam diagnosa. *EMR* juga dapat membantu memantau dan menganalisis data kesehatan pasien, yang dapat digunakan untuk meningkatkan pengambilan keputusan klinis dan memberikan perawatan yang lebih efektif dan terkoordinasi.

Banyaknya vendor software yang berbondong-bondong mulai memberikan layanan yang cepat dan akurat pada aplikasi data yang besar, dan seiringnya waktu dalam penggunaan data yang semakin besar diperlukan kompleksitas data yang beragam baik yang terstruktur maupun tidak terstruktur, sistem manajemen basis data konvensional ini masih belum menjawab tantangan secara efektif.

Kualitas pelayanan dalam menentukan memberikan Tindakan medis dan obat dengan menerapkan *EMR* akan berdampak positif dan proses semakin cepat terutama keluarga pasien akan merasakan juga dalam penerapan menggunakan beralihnya secara manual dengan tersistem.

Penerapan *Elektronik Medical Record* ini mungkin terpopuler sekitar 5 tahun kebelakang ini, dahulu banyak rumah sakit meskipun sudah menggunakan aplikasi SIM RS, tetapi penggunaan *elektronik medical record* masih manual.

Saat ini dapat kita ketahui Bersama bahwa ada dua jenis database yaitu database relasional dan database *non relational*. Database relasional mulai dikenal sejak tahun 1970 an model relasional menawarkan cara yang matematis dalam menyusun, menyimpan, dan

menggunakan data sedangkan. database non relasional jenis database yang dirancang untuk mengatasi masalah skala dan kompleksitas data yang tidak dapat ditangani dengan baik oleh database relasional. Database *non-relasional* memungkinkan pengguna untuk menyimpan dan mengakses data dalam format yang lebih fleksibel dan terukur (Rao et al., 2022), serta memungkinkan kinerja yang lebih tinggi pada data yang sangat besar dan kompleks. (Kontopoulos et al., 2022) (Slabinoha et al., 2022) .

Database non-relasional memiliki keunggulan dalam hal skalabilitas, fleksibilitas, dan performa yang lebih baik dalam mengelola data yang sangat besar dan kompleks (Lo et al., 2022), seperti data yang dihasilkan oleh aplikasi web, *IoT* (*Internet of Things*), dan Big Data (Mladenova & Valova, 2022). Namun, database non-relasional juga memiliki kekurangan, seperti kurangnya dukungan untuk transaksi *ACID* dan kurangnya standar dalam bahasa query, yang mempersulit pengolahan data secara kompleks (Praschl et al., 2022).

Ada bermacam-macam database *NoSQL* yang sering kita jumpai yaitu *Redis, MongoDB, Couchbase, Cassandra*, dan *Hbase*. Salah satu database *NoSQL* yang populer saat ini adalah MongoDB, database ini merupakan basis data yang menggunakan model penyimpanan data berorientasi dokumen (Sinaga et al., 2023) (Antas et al., 2022) (Diaz Erazo et al., 2022) (Winaya & Ashari, 2016), mongodb disimpan biasanya dengan format *JSON* atau *Binary JSON* (Yue, 2022) (M. Z. Khan et al., 2022) (W. Khan et al., 2022) (Raif Deari, 2018) (Kusumawardhana et al., 2018), mongodb menyediakan pendukung kecepatan database agar tetap optimal. (Danny Kriestanto, 2017) (Uzunbayir, 2022)

Penelitian tentang mengubah database *NoSQL* menjadi database relasional dilakukan oleh (Anchalia et al., 2022), penelitian ini menggunakan konversi database MongoDB ke Database MySQL dengan peningkatan data sebesar 75,33%. Selanjutnya oleh (Sinaga et al., 2023), penelitian ini menggunakan perbandingan kinerja insert pada database *NoSql* MongoDB dengan database *NoSql* Elasticsearch, didapat bahwa database MongoDB 42,5 kali lebih cepat. Selanjutnya oleh (Maharani, 2022), penelitian ini bertujuan untuk mengetahui kelebihan dan kekurangan basis data MongoDB, bahwa database MongoDB memiliki format berupa file *JSON* yang disebut *BSON*, format ini akan mendapatkan proses lebih cepat karena data berbasis dokumen. Selanjutnya oleh (Riki Ramdani Saputra, 2023), penelitian ini bertujuan untuk mengetahui optimasi database dengan menggunakan metode *NoSql*, bahwa didapat kinerja database transaksi dapat ditekan hingga 5%. Selanjutnya oleh (Apriliyanto et al., 2020), penelitian ini bertujuan untuk mengetahui waktu yang dapat dalam proses penyimpanan data dengan jumlah 500 data didapat bahwa database *NoSQL* memiliki kecepatan 4 kali lipat lebih cepat dibandingkan database SQL Server.

II. TINJAUAN PUSTAKA

Penelitian ini bertujuan untuk menganalisa performance menggunakan konversi database MongoDB ke Database MySQL dengan lima dataset yaitu aksesori ponsi, took buku, database universitas, restoran dan toko, bahwa proses input dengan format JSON dengan menggunakan database MySQL meningkat sebesar 75,33% (Anchalia et al., 2022)

Penelitian selanjutnya tentang perbandingan kinerja insert pada database *NoSql* MongoDB dengan database *NoSql* Elasticsearch, didapat bahwa database MongoDB 42,5 kali lebih cepat dibandingkan Elasticsearch (Sinaga et al., 2023).

Penelitian selanjutnya bertujuan untuk mengetahui kelebihan dan kekurangan basis data MongoDB, didapat bahwa MongoDB memiliki format berupa file *JSON* yang disebut *BSON*, format ini akan menyebabkan proses lebih cepat karena mampu memcached dan model data berbasis dokumen. (Maharani, 2022).

Penelitian selanjutnya bertujuan untuk mengetahui optimasi database dengan menggunakan metode *NoSql*, didapat bahwa kinerja database transaksi dapat ditekan hingga 5% (Riki Ramdani Saputra, 2023) .

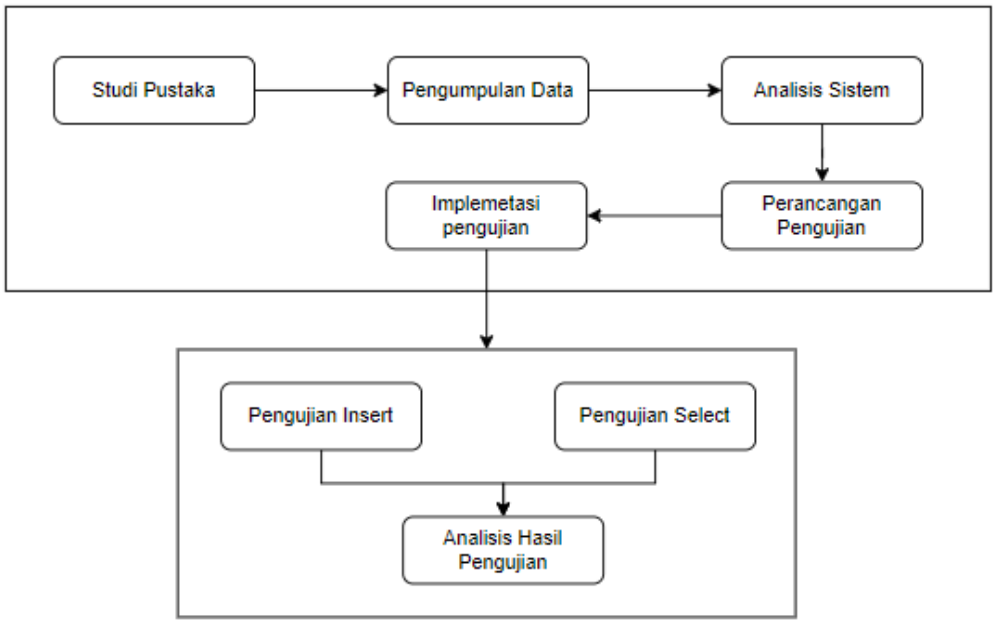
 DOI : https://doi.org/10.30646/sinus.v21i2.746

Penelitian selanjutnya bertujuan untuk mengetahui waktu yang dapat dalam proses penyimpanan data dengan jumlah 500 data didapat bahwa database *NoSQL* memiliki kecepatan 4 kali lipat lebih cepat dibandingkan database SQL Server. (Apriliyanto et al., 2020)

III. METODE PENELITIAN

Metode penelitian ini menggunakan metodologi kuantitatif. Adapun yang dimaksud dengan penelitian kuantitatif adalah adalah jenis penelitian yang mengumpulkan data berupa angka dan menggunakannya untuk menguji, mengidentifikasi pola, dan membuat generalisasi yang lebih luas. Penelitian ini berfokus pada pengumpulan data kuantitatif yang dapat diukur dan dianalisis menggunakan metode statistik.

Penelitian kuantitatif menggunakan pendekatan ilmiah yang sistematis dan objektif untuk mengumpulkan data, menganalisisnya, dan menyimpulkan temuan yang dapat diterapkan secara umum. Tujuan utama penelitian kuantitatif adalah untuk mengidentifikasi hubungan sebab-akibat, menguji hipotesis, memprediksi fenomena, dan membuat generalisasi yang berlaku dalam populasi yang lebih besar, alur metode penelitian ini ditunjukkan pada gambar 1



Gambar 1. Tahapan Penelitian

Berdasarkan gambar 1 tahapan penelitian yang telah digambarkan diatas, maka dapat diuraikan pembahasan masing-masing tahap dalam tahapan tersebut adalah sebagai berikut:

1. Studi Pustaka

   Studi pustaka merujuk pada proses pengumpulan dan penelaahan terhadap sumber-sumber pustaka yang relevan untuk mendapatkan pemahaman yang lebih baik tentang topik atau masalah tertentu. Ini melibatkan membaca, menganalisis, dan mensintesis informasi dari berbagai sumber yang berkaitan, seperti buku, jurnal ilmiah, artikel, laporan penelitian, atau sumber-sumber digital lainnya.Tujuan dari studi pustaka adalah untuk mendapatkan pemahaman yang komprehensif tentang topik yang sedang diteliti atau dibahas. Dengan melakukan studi pustaka, peneliti atau penulis dapat mengidentifikasi penelitian atau pemikiran sebelumnya yang relevan dengan topik yang sedang diteliti.

   a. Memahami konsep-konsep kunci dan kerangka teoritis yang terkait dengan topik.

b. Mengevaluasi dan membandingkan pendekatan yang telah diambil dalam penelitian sebelumnya.
c. Mengidentifikasi kekosongan penelitian atau peluang untuk penelitian lebih lanjut.
d. Mendapatkan dasar pengetahuan yang kuat untuk mendukung argumen atau klaim yang dibuat dalam penelitian atau tulisan.

2. Identifikasi Masalah

Identifikasi Masalah merupakan langkah awal yang dilakukan dalam penelitian ini. Pada tahap mengidentifikasi masalah dimaksudkan agar dapat memahami masalah yang akan diteliti, sehingga dalam tahap analisis dan perancangan tidak keluar dari permasalahan yang diteliti

3. Analisis Sistem

Pada tahap ini penulis menganalisis dan membuat rencana sistem elektronik medical record menggunakan pemodelan *UML* (*Unified Modeling Language*) dengan Langkah-langkah sebagai berikut:

a. Perencanaan awal dan Tujuan Pemodelan
b. Identifikasi sistem yang digunakan saat ini
c. Membuat pemodelan *UML* pada aplikasi *ERM*
d. Membuat Protipe *ERM*

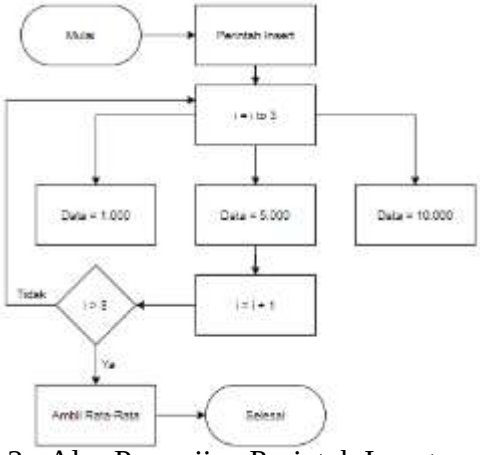
4. Perancangan Sistem

Pada tahap ini kita merancang sistem yang baru, penulis menggunakan metode pengembangan sistem dengan model Prototyping system pada aplikasi *ERM*

5. Hasil Pengujian

Pada tahapan ini akan diketahui hasil response time pada aplikasi *ERM* dengan menggunakan database *Postgresql* dan *MongoDB*.

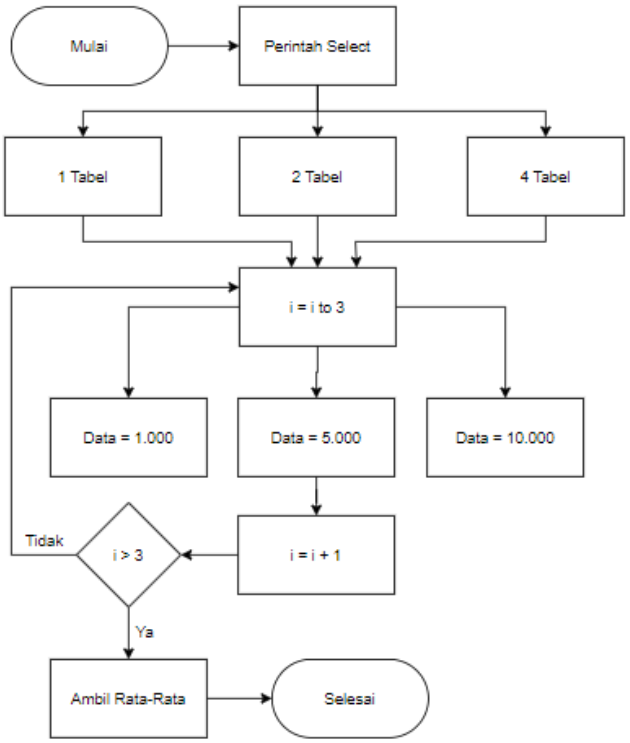
Jenis penelitian ini menggunakan metode untuk menguji kinerja kecepatan kedua database yaitu RDBMS dengan menggunakan studi kasus *PostgreSQL* dan *NoSQL* dengan menggunakan studi kasus mongodb, dengan mengaplikasikan perintah select, dan insert pada aplikasi Elektronik Medical Record (EMR) dengan 3 jenis skenario data, data sebanyak 1.000, 5.000, dan data sebanyak 10.000. setiap skenario dilakukan secara bersamaan sehingga didapat respon time.



Gambar 2. Alur Pengujian Perintah Insert

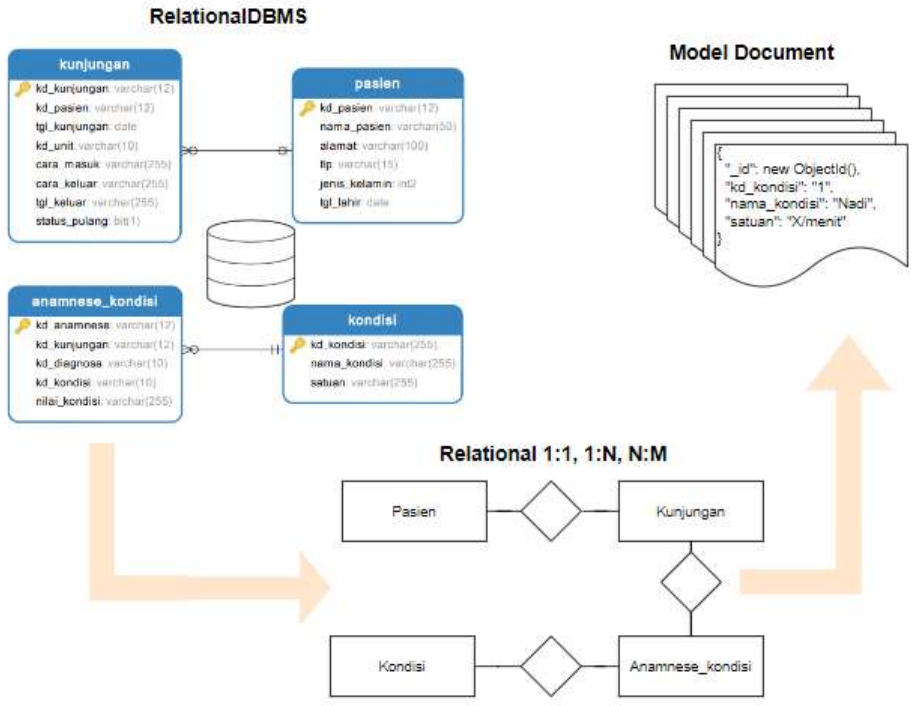
Pada Gambar 2 dapat kita lihat alur pengujian perintah insert dimulai dari pengujian 1.000 record, 5.000, kemudian terakhir 10.000 record. Tiap-tiap record diuji lagi sebanyak 3 kali kemudian diambil waktu rata-ratanya.

 DOI : https://doi.org/10.30646/sinus.v21i2.746



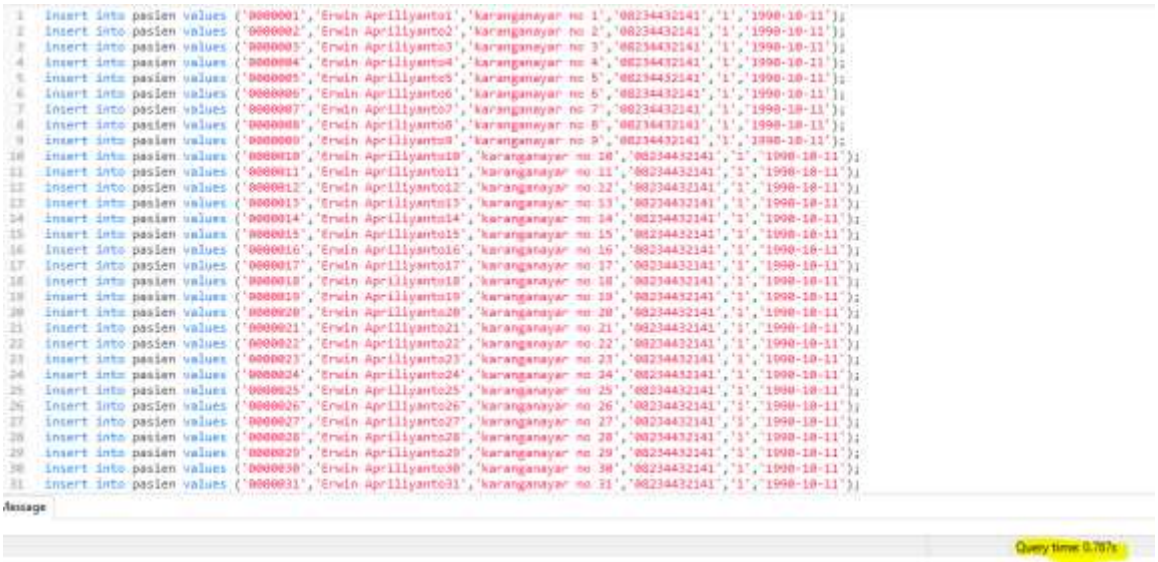
Gambar 3. Alur Pengujian Perintah Select

Pada Gambar 3 dapat kita lihat pengujian perintah select yaitu dengan 1 tabel, 2 tabel dan 4 tabel, setelah itu diuji sebanyak tiga kali dan diambil rata-rata waktunya.



Gambar 4. Alur Proses Transformasi Skema Database (Winaya & Ashari, 2016)

Pada gambar 4 mengilustrasikan jenis DBMS dengan *NoSql*, untuk DMS menggunakan relasi antar tabel sedangkan untuk *NoSql* menggunakan konsep document



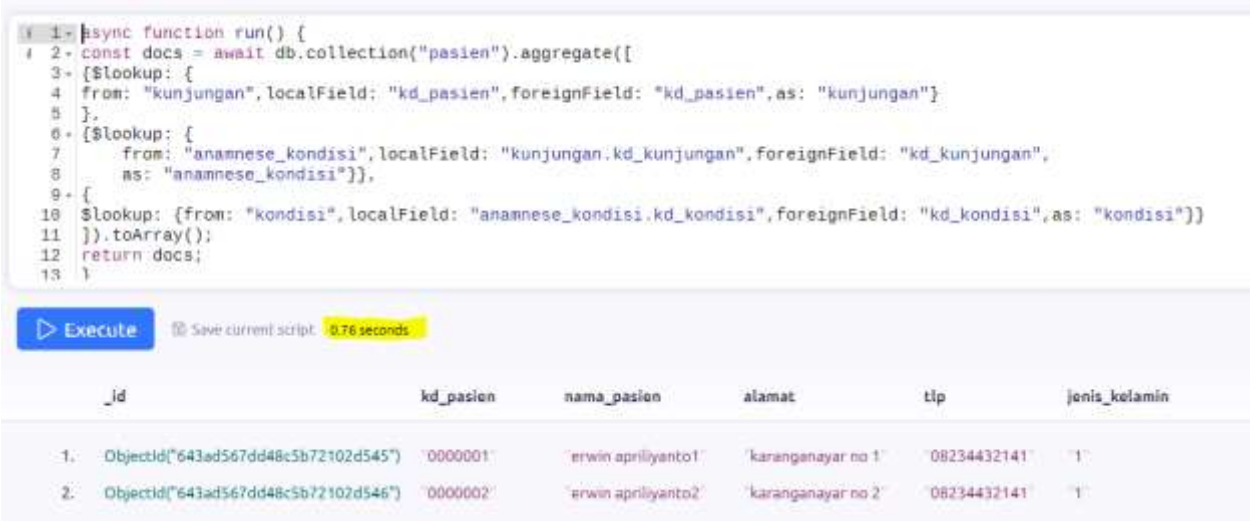
Gambar 5. Perintah Query Insert PostgreSQL

Pada gambar 5 syntax untuk menjalankan perintah insert atau memasukkan data pada tabel pasien

Gambar 6. Perintah Query Insert MongoDB

Pada gambar 6 syntax untuk menjalankan perintah insert atau memasukkan data ke dokumen pasien pada MongoDB

 DOI : https://doi.org/10.30646/sinus.v21i2.746



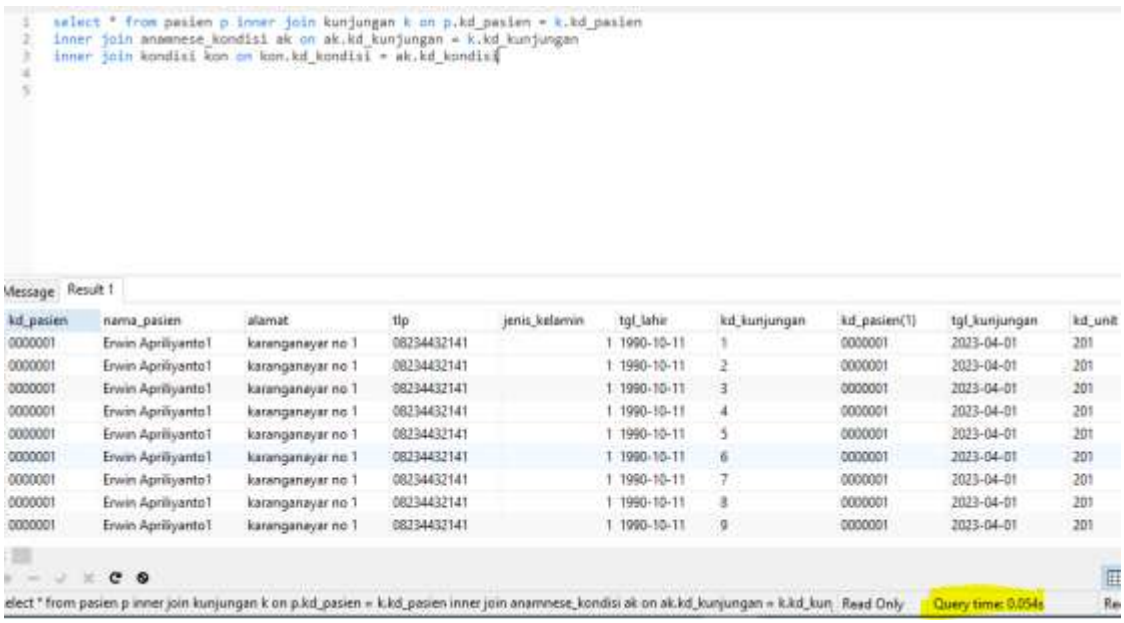
```
async function run() {
const docs = await db.collection("pasien").aggregate([
{$lookup: {
from: "kunjungan",localField: "kd_pasien",foreignField: "kd_pasien",as: "kunjungan"}
},
{$lookup: {
    from: "anamnese_kondisi",localField: "kunjungan.kd_kunjungan",foreignField: "kd_kunjungan",
    as: "anamnese_kondisi"}},
{
$lookup: {from: "kondisi",localField: "anamnese_kondisi.kd_kondisi",foreignField: "kd_kondisi",as: "kondisi"}}
]).toArray();
return docs;
}
```

Execute | Save current script | 0.76 seconds

	_id	kd_pasien	nama_pasien	alamat	tlp	jenis_kelamin
1.	ObjectId("643ad567dd48c5b72102d545")	0000001	erwin apriliyanto1	karanganayar no 1	08234432141	1
2.	ObjectId("643ad567dd48c5b72102d546")	0000002	erwin apriliyanto2	karanganayar no 2	08234432141	1

Gambar 7. Perintah Query Join pada MongoDB

Pada gambar 7 adalah syntax dengan konsep inner join dan left join dengan menggunakan konsep json *NoSQL* pada mongoDB dengan scenario 4 tabel diantara, tabel pasien, tabel kunjungan, tabel anamnesis kondisi dan tabel kondisi.



```
select * from pasien p inner join kunjungan k on p.kd_pasien = k.kd_pasien
inner join anamnese_kondisi ak on ak.kd_kunjungan = k.kd_kunjungan
inner join kondisi kon on kon.kd_kondisi = ak.kd_kondisi
```

Message | Result 1

kd_pasien	nama_pasien	alamat	tlp	jenis_kelamin	tgl_lahir	kd_kunjungan	kd_pasien(1)	tgl_kunjungan	kd_unit
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	1	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	2	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	3	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	4	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	5	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	6	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	7	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	8	0000001	2023-04-01	201
0000001	Erwin Apriliyanto1	karanganayar no 1	08234432141	1	1990-10-11	9	0000001	2023-04-01	201

elect * from pasien p inner join kunjungan k on p.kd_pasien = k.kd_pasien inner join anamnese_kondisi ak on ak.kd_kunjungan = k.kd_kun | Read Only | Query time: 0.054s

Gambar 8. Perintah Query Join pada PostgreSQL

Pada gambar 8 adalah menampilkan data pasien berkunjungan di rumah sakit tersebut dengan melibatkan 4 tabel dengan menggunakan syntax inner join dan left join maka didapat dengan waktu 0.054 second.

IV. HASIL DAN PEMBAHASAN

4.1 Karakteristik Data

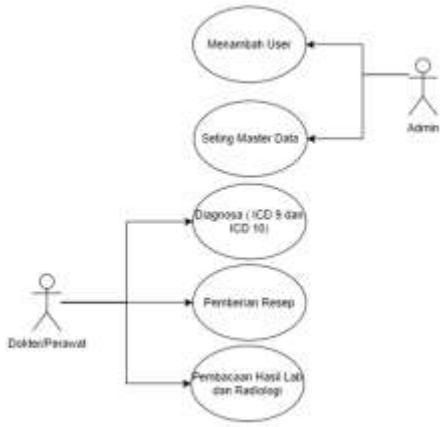
Referensi data didapat dari seorang pakar, dan link resmi kemenkes. Selanjutnya data dilakukan import ke database yang sudah disiapkan, ini salah satu contoh data yang diimpor ke database tertera pada tabel 1.

Tabel 1 Data Diagnosa

ICD 10	Parent	Description ICD 10
A00	A00-A09	Cholera
A00.0	A00-A09	Cholera due to Vibrio cholerae 01, biovar cholerae
A00.1	A00-A09	Cholera due to Vibrio cholerae 01, biovar eltor
A00.9	A00-A09	Cholera, unspecified
A01	A00-A09	Typhoid and paratyphoid fevers
A01.0	A00-A09	Typhoid fever

4.2 Diagram Use Case

Diagram Use Case (diagram kasus penggunaan) adalah salah satu jenis diagram UML yang digunakan untuk menggambarkan interaksi antara pengguna atau aktor dengan sistem. Diagram ini membantu dalam pemahaman yang lebih baik tentang fungsionalitas sistem dari perspektif pengguna.Identifikasi pengguna atau entitas luar lainnya yang berinteraksi dengan sistem. Aktor dapat berupa pengguna manusia, perangkat keras, sistem lain, atau entitas eksternal lainnya yang berperan dalam interaksi dengan sistem. Diagram Use Case membantu dalam memahami perspektif pengguna dalam menggunakan sistem. Ini juga dapat digunakan sebagai dasar untuk mengidentifikasi kebutuhan fungsional sistem, merancang skenario pengujian, dan menyusun dokumentasi sistem yang lebih rinci.



Gambar 9. Perintah Query Join pada PostgreSQL

Berdasarkan gambar 9 diagram use case di atas maka alur proses aplikasi *ERM* sebagai berikut: a.) Pengguna dalam system ini dibagi menjadi 2 kelompok yaitu admin dan dokter/perawat b.) Admin dapat mengelola master data missal berupa menambahan user, seting diagnose c.) pengguna dokter dapat melakukan proses pemberian diagnose pada pasien, pemberian resep, pemerikasaan hasil laboratorium dan radiologi.

Tabel 2 Perbandingan Respontime dengan Perintah Insert

Jumlah Data	Skenario	Waktu		Rata-rata	
		PostgreSQL	NoSql	PostgreSQL	NoSql
1.000	1	7.800 ms	2. ms	7.800 ms	2. ms
	2	7.800 ms	2. ms		
	3	7.800 ms	2. ms		
5.000	1	4.710 ms	10 ms	4.710 ms	10 ms
	2	4.710 ms	10 ms		
	3	4.710 ms	10 ms		
10.000	1	8.540 ms	24 ms	8.540 ms	24 ms
	2	8.540 ms	24 ms		
	3	8.540 ms	24 ms		

 DOI : https://doi.org/10.30646/sinus.v21i2.746

Pada tabel 2 untuk jumlah data 1.000 dengan skenario ke 1 waktu yang didapat untuk menjalankan perintah insert sebesar 7.800 ms untuk database DBMS PostgreSQL sedangkan waktu yang didapat untuk menjalankan perintah insert pada database *NoSQL* sebesar 2 ms.

Tabel diatas dapat diolah menggunakan rumus perbandingan, sehingga didapat jumlah perbandingan respontime antara database *PostgreSql* dengan Database *NoSql* perhitungannya sebagai berikut

X_1 = Variabel *PostgreSQL*

X_2= Waktu Response untuk Database *PostgreSQL*

Y_1= Variabel *NoSQL*

Y_2= Waktu Response untuk Database *NoSQL*

$\frac{X_1}{Y_1} = \frac{X_2}{Y_2}$ ➔ $\frac{X_1}{Y_1} = \frac{7.800\ ms}{2\ ms}$ ➔X_1 = 3.900 ms Y_1

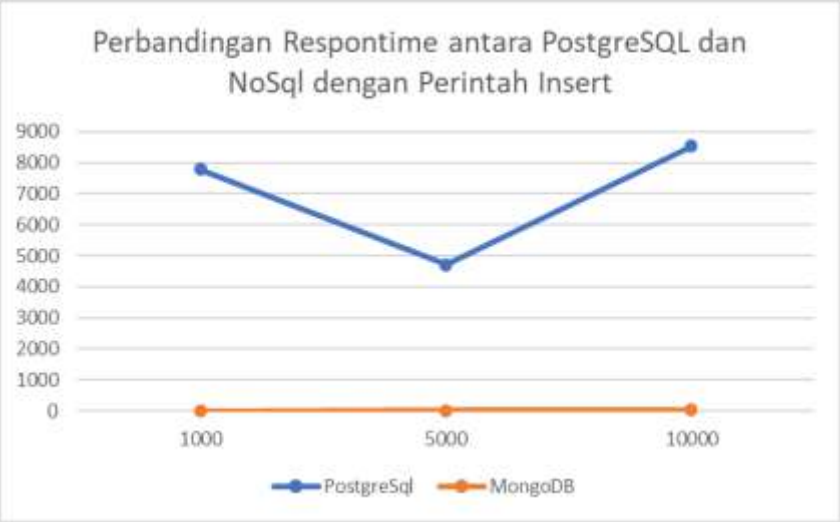
Jadi untuk perintah *insert* pada database *PostgreSQL* dengan data 1.000 lebih lambat 3.900 dari database *NoSql* atau database *NoSql* Lebih cepat 3.900 kali dibandingkan database *PostgreSql*.

$\frac{X_1}{Y_1} = \frac{X_2}{Y_2}$ ➔$\frac{X_1}{Y_1} = \frac{4.700\ ms}{10\ ms}$ ➔X_1 = 470 ms Y_1

Jadi untuk perintah *insert* didapat hasil bahwa database *PostgreSQL* untuk data 5.000 lebih lambat 470 dari database *NoSql* atau database *NoSql* Lebih cepat 470 kali dibandingkan database *PostgreSql*.

$\frac{X_1}{Y_1} = \frac{X_2}{Y_2}$ ➔ $\frac{X_1}{Y_1} = \frac{8.540\ ms}{24\ ms}$ ➔X_1 = 355 ms Y_1

Jadi untuk perintah *insert* didapat hasil bahwa database *PostgreSQL* untuk data 10.000 lebih lambat 355 dari database *NoSql* atau database *NoSql* Lebih cepat 355 kali dibandingkan database *PostgreSql*.



Gambar 10. Perbandingan Respontime antara Perintah Insert

Pada gambar 10 menunjukkan hasil dari pengujian perintah Sql insert pada database Elektronik Medical Record, dengan menggunakan 1000 record, 5000 record, 1.0000 record

Tabel 3 Perbandingan Perintah Select dengan 1 Tabel

Jumlah Data	Skenario	Waktu		Rata-rata	
		PostgreSQL	NoSql	PostgreSQL	NoSql
1.000	1	62 ms	3. ms	37 ms	3. ms
	2	32 ms	2. ms		
	3	17 ms	4. ms		
5.000	1	51 ms	12 ms	67 ms	6 ms
	2	55 ms	2 ms		
	3	95 ms	4 ms		
10.000	1	94 ms	18 ms	87 ms	13 ms
	2	127 ms	8 ms		
	3	39 ms	13 ms		

Pada tabel 3 untuk jumlah data 1.000 dengan skenario ke 2 waktu yang didapat untuk menjalankan perintah select sebesar 32 ms untuk database DBMS PostgreSQL sedangkan waktu yang didapat untuk menjalankan perintah select pada database *NoSQL* sebesar 2 ms.

Tabel diatas dapat diolah menggunakan rumus perbandingan, sehingga didapat jumlah perbandingan respontime antara database *PostgreSql* dengan Database *NoSql* perhitungannya sebagai berikut

X_1 = Variabel *PostgreSQL*

X_2= Waktu Response untuk Database *PostgreSQL*

Y_1= Variabel *NoSQL*

Y_2= Waktu Response untuk Database *NoSQL*

$\frac{X_1}{Y_1} = \frac{X_2}{Y_2}$ ➔$\frac{X_1}{Y_1} = \frac{37\ ms}{3\ ms}$ ➔X_1 = 12 ms Y_1

Jadi untuk perintah *select* didapat hasil bahwa database *PostgreSQL* untuk data 1.000 lebih lambat 12 dari database *NoSql* atau database *NoSql* Lebih cepat 12 kali dibandingkan database PostgreSql.

Gambar 11. Perbandingan Perintah Select dengan 1 Tabel

Gambar 11. Grafik Perbandingan Perintah Select 1 Tabel Pada MongoDB dan PostgreSQL. Melakukan ujicoba ini untuk menguji kecepatan dalam memproses data dengan perintah select untuk 1 tabel dalam waktu yang bersamaan, di bawah ini adalah daftar ujicoba penelitian kami dalam format tabel dan grafik.

 DOI : https://doi.org/10.30646/sinus.v21i2.746

Tabel 4 Perbandingan Perintah Select dengan Join 2 Tabel

Jumlah Data	Skenario	Waktu		Rata-rata	
		PostgreSQL	NoSql	PostgreSQL	NoSql
1.000	1	350 ms	730 ms	130 ms	683. ms
	2	530 ms	640 ms		
	3	550 ms	680 ms		
5.000	1	85 ms	4260 ms	130 ms	4.377 ms
	2	130 ms	4400 ms		
	3	85 ms	4470 ms		
10.000	1	68 ms	6.610 ms	283 ms	6.663 ms
	2	74 ms	6.840 ms		
	3	103 ms	6.540 ms		

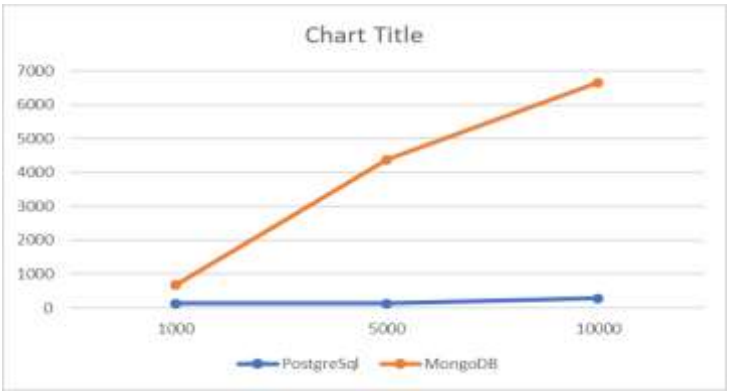
Pada tabel 4 untuk jumlah data 1.000 dengan skenario ke 3 waktu yang didapat untuk menjalankan perintah insert sebesar 550 ms untuk database DBMS PostgreSQL sedangkan waktu yang didapat untuk menjalankan perintah insert pada database *NoSQL* sebesar 680 ms.

Tabel diatas dapat diolah menggunakan rumus perbandingan, sehingga didapat jumlah perbandingan respontime antara database *PostgreSql* dengan Database *NoSql* perhitungannya sebagai berikut

X_1 = Variabel *PostgreSQL*
X_2= Waktu Response untuk Database *PostgreSQL*
Y_1= Variabel *NoSQL*
Y_2= Waktu Response untuk Database *NoSQL*

$$\frac{X_1}{Y_1} = \frac{X_2}{Y_2} \rightarrow \frac{X_1}{Y_1} = \frac{130\ ms}{683\ ms} \rightarrow Y_1 = 5 \text{ ms } X_1$$

Jadi untuk perintah *select* 2 tabel didapat hasil bahwa database *PostgreSQL* untuk data 1.000 lebih cepat 5 dari database *NoSql* atau database *NoSql* Lebih lambat 5 kali dibandingkan database *PostgreSql*.



Gambar 12. Perbandingan Perintah Select dengan 2 Tabel

Gambar 12. Menunjukkan Grafik Perbandingan Perintah Select dengan join antar 2 Tabel Pada MongoDB dan PostgreSQL. Melakukan ujicoba ini untuk menguji kecepatan dalam memproses data dengan perintah select untuk 2 tabel dalam waktu yang bersamaan, di bawah ini adalah daftar ujicoba penelitian kami dalam format tabel dan grafik.

Tabel 5 Perbandingan Perintah Select dengan 4 Tabel

Jumlah Data	Skenario	Waktu		Rata-rata	
		PostgreSQL	NoSql	PostgreSQL	NoSql
1.000	1	61 ms	2.140. ms	59,33 ms	2.087 ms
	2	60 ms	2.115. ms		
	3	57 ms	2.008. ms		
5.000	1	62 ms	8.560 ms	60,67 ms	8.398 ms
	2	61 ms	8.410 ms		
	3	59 ms	8.225 ms		
10.000	1	66 ms	94.830 ms	62,33 ms	8.618 ms
	2	60 ms	86.440 ms		
	3	61 ms	77.270 ms		

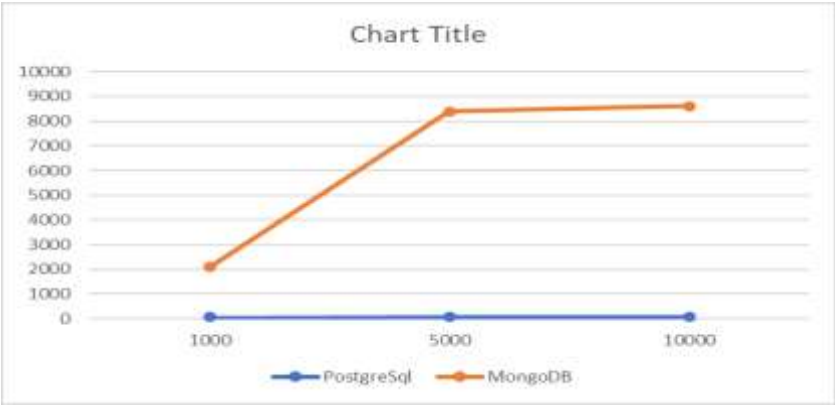
Pada tabel 5 untuk jumlah data 5.000 dengan skenario ke 1 waktu yang didapat untuk menjalankan perintah insert sebesar 62 ms untuk database DBMS PostgreSQL sedangkan waktu yang didapat untuk menjalankan perintah insert pada database *NoSQL* sebesar 8.560 ms.

Tabel diatas dapat diolah menggunakan rumus perbandingan, sehingga didapat jumlah perbandingan respontime antara database *PostgreSql* dengan Database *NoSql* perhitungannya sebagai berikut

X_1 = Variabel *PostgreSQL*
X_2= Waktu Response untuk Database *PostgreSQL*
Y_1= Variabel *NoSQL*
Y_2= Waktu Response untuk Database *NoSQL*

$$\frac{X_1}{Y_1} = \frac{X_2}{Y_2} \rightarrow \frac{X_1}{Y_1} = \frac{59{,}33\ ms}{2.087\ ms} \rightarrow Y_1 = 35 \text{ ms } X_1$$

Jadi untuk perintah *select* 4 tabel didapat hasil bahwa database *PostgreSQL* untuk data 1.000 lebih cepat 35 dari database *NoSql* atau database *NoSql* Lebih lambat 35 kali dibandingkan database *PostgreSql*.



Gambar 13. Perbandingan Perintah Select dengan 4 Tabel

V. KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Dapat ditarik kesimpulan bahwa perintah Insert dengan 1.000 record data akan mendapatkan hasil database *NoSQL* lebih cepat 3.900 kali dibandingkan database

PostgreSQL, sedangkan dengan 5.000 record data database *NoSQL* 471 kali lebih cepat, sedangkan dengan 10.000 record data database *NoSQL* 355 kali lebih cepat, dan perintah *select* untuk 1 tabel database *NoSQL* 8 kali lebih cepat dari database *PostgreSQL*

5.2 SARAN

Penggunaan uji coba kecepatan dengan skenario maksimal 100.000 data, sehingga dapat mengetahui response time database yang sangat besar dengan menggunakan 10 kali uji coba setiap satu skenario.

DAFTAR PUSTAKA

Anchalia, A., Paudel, A., Sanjeetha, R., & Kakati, A. (2022). Transforming NoSQL Database to Relational Database: An Algorithmic Approach. *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, 1–9. https://doi.org/10.1109/GCAT55367.2022.9972168

Antas, J., Rocha Silva, R., & Bernardino, J. (2022). Assessment of SQL and NoSQL Systems to Store and Mine COVID-19 Data. *Computers*, *11*(2), 29. https://doi.org/10.3390/computers11020029

Apriliyanto, E., Laksmita, N. C., Pandu, I. W., & Rini, K. (2020). Comparison of NoSQL Database Performance with SQL Server Database on Online Airplane Ticket Booking. *Indonesian Journal of Applied Informatics*, *4*(2), 64. https://doi.org/10.20961/ijai.v4i2.38956

Chopade, M. R. M., & Dhavase, N. S. (2017). Mongodb, couchbase: Performance comparison for image dataset. *2017 2nd International Conference for Convergence in Technology (I2CT)*, 255–258. https://doi.org/10.1109/I2CT.2017.8226131

Danny Kriestanto, A. B. A. (2017). Implementasi Website Pencarian Kos Dengan Nosql. *Danny Kriestanto, Alif Benden Arnado*, *2*. https://doi.org/http://dx.doi.org/10.26798/jiko.v2i2.66

Diaz Erazo, A. D., Raul Morales Morales, M., Pineda Chavez, V. K., & Leonardo Morales Cardoso, S. (2022). Comparative Analysis of performance for SQL and NoSQL Databases. *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–14. https://doi.org/10.23919/CISTI54924.2022.9820292

Khan, M. Z., Zaman, F. U., Adnan, M., Imroz, A., Rauf, M. A., & Phul, Z. (2022). Comparative Case Study: An Evaluation of Performance Computation between SQL and NoSQL Database. *2022 Sindh Journal of Headways in Software Engineering*, *1*. http://sjhse.smiu.edu.pk/sjhse/index.php/SJHSE/article/view/42

Khan, W., Kumar, T., Cheng, Z., Raj, K., Roy, A. M., & Luo, B. (2022). *SQL and NoSQL Databases Software architectures performance analysis and assessments -- A Systematic Literature review*. 57. http://arxiv.org/abs/2209.06977

Kontopoulos, I., Makris, A., Xyalis, S. N., & Tserpes, K. (2022). Benchmarking moving object functionalities of DBMSs using real-world spatiotemporal workload. *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, 388–393. https://doi.org/10.1109/MDM55031.2022.00087

Kusumawardhana, P. M., Mochammad Hannats Hanafi Ichsan, & Rakhmadhany Primananda. (2018). Implementasi Penyimpanan Data Sensor Nirkabel dengan MongoDB pada Lingkungan IOT Menggunakan Protokol MQTT. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/2291

Lo, T.-C., Tao, C.-Y., Chang, J.-B., & Shieh, C.-K. (2022). Performance Comparison of Containerized HBase Clusters on Kubernetes. *2022 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, 1–7.

https://doi.org/10.1109/RASSE54974.2022.9989814

Maharani, A. (2022). Perancangan Data Base Kasir Dan Persediaan Barang Menggunakan Mongodb. *Jurnal Data Mining Dan Sistem Informasi*, *3*(1), 32. https://doi.org/10.33365/jdmsi.v3i1.1941

Mladenova, T., & Valova, I. (2022). Performance Study of MySQL and MongoDB for IoT Data Processing and Storage. *2022 International Conference Automatics and Informatics (ICAI)*, 60–63. https://doi.org/10.1109/ICAI55857.2022.9960134

Praschl, C., Pritz, S., Krauss, O., & Harrer, M. (2022). A Comparison Of Relational, NoSQL and NewSQL Database Management Systems For The Persistence Of Time Series Data. *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, 1–6. https://doi.org/10.1109/ICECCME55909.2022.9988333

Raif Deari, Xhemal Zenuni, Jaumin Ajdari, Florije Ismaili, B. R. (2018). Analysis And Comparison Of Document-Based Databases With Sql Relational Databases: Mongodb Vs Mysql. *Proceedings of the International Conference on Information Technologies (InfoTech-2018)*. http://infotech-bg.com/sites/default/files/2018/B04.pdf

Rao, A., Khankhoje, D., Namdev, U., Bhadane, C., & Dongre, D. (2022). Insights into NoSQL databases using financial data: A comparative analysis. *Procedia Computer Science*, *215*, 8–23. https://doi.org/10.1016/j.procs.2022.12.002

Riki Ramdani Saputra. (2023). Optimalisasi database transaksi telekomunikasi dengan nosql memcached. *(2023): jatisi (Jurnal Teknik Informatika Dan Sistem Informasi)*, *10*. https://doi.org/https://doi.org/10.35957/jatisi.v10i1.3228

Sinaga, T. L., Charibaldi, N., & Cahyana, N. H. (2023). Perbandingan Waktu Respon Aplikasi Database NoSQL Elasticsearch dan MongoDB pada Pengujian Operasi CRUD. *JISKA (Jurnal Informatika Sunan Kalijaga)*, *8*(1), 22–35. https://doi.org/10.14421/jiska.2023.8.1.22-35

Slabinoha, M., Melnychuk, S., Manuliak, I., & Pashkovskyi, B. (2022). Comparative analysis of embedded databases performance on single board computer systems using Python. *2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT)*, 222–225. https://doi.org/10.1109/CSIT56902.2022.10000475

Uzunbayir, S. (2022). Relational Database and NoSQL Inspections using MongoDB and Neo4j on a Big Data Application. *2022 7th International Conference on Computer Science and Engineering (UBMK)*, 148–153. https://doi.org/10.1109/UBMK55850.2022.9919589

Winaya, I. G., & Ashari, A. (2016). Transformasi Skema Basis Data Relasional Menjadi Model Data Berorientasi Dokumen pada MongoDB. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *10*(1), 47. https://doi.org/10.22146/ijccs.11188

Yue, K.-B. (2022). Assuring Referential Integrity in Blockchain Applications. *2022 Fourth International Conference on Blockchain Computing and Applications (BCCA)*, 47–54. https://doi.org/10.1109/BCCA55292.2022.9921976