Analisis Performa Neo4j, MongoDB, dan PostgreSQL sebagai *Database* Manajemen *Big Data* Pemilu 2019

Linggis Galih Wiseso[1], Mahmud Imrona[2], Andry Alamsyah[3]

[1,2,]Fakultas Informatika, Universitas Telkom, Bandung
[3]Fakultas Ekonomi Bisnis, Universitas Telkom, Bandung
[1]galihwiseso@students.telkomuniversity.ac.id, [2]mahmudimrona@telkomuniversity.ac.id,
[3]andrya@telkomuniversity.ac.id

Abstrak

Data kini telah menjadi komoditas utama untuk suatu organisasi karena data telah mengubah cara mereka berbisnis, berkomunikasi, dan mengambil keputusan. Data yang diperoleh berupa *big data*. Salah satu organisasi yang menggunakan *big data* adalah partai politik. Mereka menggunakan *big data* untuk menganalisis kemungkinan menangnya kandidat mereka dalam pemilu. Namun di Indonesia penggunaannya belum bisa maksimal karena keterbatasan pada infrastrukturnya. Maka dari itu, untuk memaksimalkan penggunaan *big data* diperlukan sebuah manajemen penyimpanan data yang dapat menampung big data. Salah satu *database* tersebut adalah *graph database* Pada penelitian ini, penulis mengkaji tiga *tools database* yang sering digunakan yaitu PostgreSQL mewakili *relational database*, MongoDB mewakili *document-based database* dan Neo4J mewakili *graph database*. Penulis melakukan *query* searching yang memiliki klausa JOIN pada ketiga *database*. Untuk membandingkan performa antar *database* engine, penulis menggunakan notasi Big O yang merupakan notasi yang dipakai umum dalam menggambarkan kompleksitas suatu sistem memproses data. Kesimpulan dari penelitian ini adalah MongoDB paling bagus untuk mengelola big data pemilu. Dimana MongoDB memiliki kompleksitas O(1). Hal ini terjadi karena tipe *database document based* seperti MongoDB memudahkan penyimpanan dan query datanya ke dalam satu *collection* yang sama.

Kata kunci : big data, postgresql, mongodb, neo4j, big o, performa *database*

Abstract

Data are now a major commodities for an organization because it changed their way of business, communicates and makes decisions. The data obtained is in form of big data. One type of organization that rely on big data is political party.They use it to analyze the likelihood of their candidate winning in the election. But in Indonesia, its has not been to utilized because of limited infrastructure. Therefore, to utilized the use of big data. It required a data storage management that can accommodate big data. One of these databases is graph database. In this study, the author examines three database tools that often used, namely PostgreSQL representing relational database, MongoDB representing document-based database and Neo4J representing graph database. The author performs searching query that has a JOIN clause on all three databases. To compare their performance, the author uses Big O Notation which is a notation that is commonly used in describing the complexity of a data processing system. The conclusion from this study is that MongoDB is very good for managing Big Data elections. Where MongoDB has complexity of O(1). This happens because document based database like MongoDB easy to store and query data in the same collection.

Keywords: big data, postgresql, mongodb, neo4j, big o, database performance

1. Pendahuluan

Latar Belakang

"Data adalah jenis kekayaan baru bangsa kita, kini data lebih berharga dari minyak," sebut Presiden Republik Indonesia Joko Widodo dalam pidato kenegaraannya di Hari Ulang Tahun ke-74 Kemerdekaan Republik Indonesia 2019 [1]. Data kini telah menjadi komoditas utama untuk suatu organisasi karena data telah mengubah cara mereka berbisnis, berkomunikasi, dan mengambil keputusan [2]. Data yang diperoleh organisasi tersebut memiliki jumlah yang sangat besar dan beragam, karena itu muncul istilah *Big Data*. *Big data* secara singkat adalah istilah yang menggambarkan *volume* data yang besar, baik data yang terstruktur maupun data yang tidak terstruktur [3]. Dengan adanya *big data*, suatu organisasi dapat menganalisa data tersebut dan hasilnya digunakan untuk pengambilan keputusan dan strategi bisnis. Salah satu organisasi yang menggunakan *big data* adalah partai politik.

Partai politik menggunakan *big data* untuk menganalisis kemungkinan menangnya kandidat mereka dalam pemilu. Di Indonesia sendiri penggunaan *big data* pada pemilu 2019 belum maksimal karena terbatasnya

infrastruktur [4]. Jika Indonesia ingin memaksimalkan penggunaan *big data*, maka perlu sebuah manajemen penyimpanan data yang dapat menampung big data. Salah satu *database* tersebut adalah *graph database*.

Graph database adalah *database management system* yang memiliki metode *create, read, update,* dan *delete* untuk penyimpanan data nya dengan menggunakan model data graf. *Graph database* memiliki kumpulan *nodes* dan *relationship* yang saling terhubung. Sehingga *graph database* ini memiliki performa yang bagus dalam fleksibilitas dan *agility* [5]. Salah satu *tools* pengelolaan *graph database* adalah Neo4j. Sebelumnya telah dilakukan studi terkait penggunan *graph database* dengan Neo4j untuk *database* manajemen *big data*. Salah satu studi tersebut melakukan analisis performa Neo4j, MongoDB, dan PostgreSQL untuk menentukan *database* yang cocok untuk sebuah aplikasi berbasis *Geographical Information System* (GIS) [6]. Pada penelitian ini, penulis mengkaji tiga *databse tools* yang sering digunakan yaitu PostgreSQL mewakili *relational database*, MongoDB mewakili *document-based database* dan Neo4J mewakili *graph database*.

Topik dan Batasan

Topik yang diangkat pada penelitian ini adalah untuk menganalisa bagaimana performa dari Neo4j, MongoDB, dan PostgreSQL sebagai *database* manajemen *big data* Pemilu 2019.

Adapun batasan pada penelitian ini adalah penulis tidak melakukan pengujian pada penggunaan memori karena terbatasnya memori laptop penulis, tidak menggunakan *index* pada ketiga *database*, dan hanya 1,000,000 data yang di-*query*.

Tujuan

Tujuan dari penelitian ini adalah untuk mengetahui *tools database* yang bagus untuk *database* manajemen *big data* Pemilu 2019 berdasarkan performanya.

Organisasi Tulisan

Dalam jurnal Tugas Akhir (TA) ini telah dibagi-bagi beberapa bagian setelah pendahuluan. Yaitu studi terkait yang berisikan studi pustaka terhadap penelitian-penelitian sebelumnya yang menjadi referensi dari penelitian TA penulis, sistem yang dibangun berisikan penjelasan mengenai sistem yang telah dibangun, evaluasi berisikan Hasil Pengujian dan Analisis Hasil Pengujian pada penelitian TA ini, kesimpulan yang memuat kesimpulan dari penelitian TA ini dan saran untuk penelitian kedepannya, daftar pustaka yang berisikan sumber referensi yang digunakan pada penelitian TA ini, dan lampiran yang berisikan detail data hasil pengujian dan *query* yang digunakan.

2. Studi Terkait

Pada penelitian yang dilakukan oleh Monika Sharma, Vishal Deep Sharma, dan Mahesm M. Bundele pada tahun 2018 melakukan analisis performa antara *Relation Database Management System* (RDBMS), *document-based* NoSQL, dan *graph-based* NoSQL untuk sebuah data *geo tagged* pada skenario GIS [6]. *Database* yang mereka analisis adalah PostgreSQL untuk RDBMS, MongoDB untuk *document based* NoSQL, dan Neo4j untuk graph *based* NoSQL. Hasil dari penelitian ini adalah MongoDB merupakan *database* yang memiliki waktu ekseskusi *query* tercepat ketimbang PostgreSQL dan Neo4j.

Penelitian selanjutnya dilakukan oleh I Nyoman Pande Wahyu Dharmawan dan Riyanarto Sarno pada tahun 2017 melakukan rekomendasi buku dengan *graph database* dan *book metadata* yang disimpan pada Neo4j [7]. Hasil dari penelitian ini didapatkan bahwa *graph database* membutuhkan waktu yang lama untuk eksekusi *query* ketimbang MySQL. Namun untuk penyimpanan rekomendasi buku ke *database* lebih cocok Neo4j karena fleksibilitasnya dalam menyimpan data.

Penelitian berikutnya dilakukan oleh Huiling Lu, Zhiguo Hong, dan Minyong Shi pada tahun 2017 melakukan analisis keterkaitan antara *key objects* pada data film menggunakan Neo4j [8]. Untuk mengetahui performa Neo4j, mereka melakukan *query* atribut pada *single node*, *query* pada hubungan antara dua *nodes* atau lebih, dan *query* mencari hubungan tersembunyi pada *nodes*. Hasil dari penelitian ini adalah Neo4j cocok untuk penanganan data yang jumlahnya besar, kompleks, dinamis, interaktif, dan *low structured*.

Penelitian terakhir dilakukan oleh Chad Vicknair, Michael Macias, Zhendong Zhao, dkk. pada tahun 2010 melakukan perbandingan antara *graph database* dan *relational database* untuk pengembangan *software* penyimpanan dan *query data provenance information* [9]. Pada penelitian ini menggunakan dua *database,* MySQL untuk *relational database* dan Neo4j untuk *graph database*. Hasil dari penelitian ini adalah *graph database* melakukan lebih baik pada *query* tipe struktural daripada *relational database*. Dalam pencarian karakter teks, *graph database* bekerja lebih signifikan daripada *relational database*.

Big data adalah istilah untuk sebuah data yang memiliki tiga karakteristik yaitu, *extremely large volumes of data*, *extremely high velocity of data*, dan *extremely wide variety of data* [10]. Dengan adanya *big data* dapat memudahkan sebuah organisasi untuk mengambil, menyimpan, mengatur, dan merubah jumlah data yang besar pada kecepatan yang tepat, waktu yang tepat, dan mendapatkan pengetahuan yang tepat [10].

Graph database adalah salah satu bagian dari *database* NoSQL yang metode *Create, Read, Update,*dan

Delete penyimpanan datanya menggunakan model data graf [5]. Sebuah *graph database* terdiri dari dua elemen, yaitu *nodes* dan *relationship* [11]. Masing-masing *nodes* merepresentasikan sebuah entitas (orang, tempat, kategori, dst.) dan *relationship* merepresentasikan bagaimana kedua *nodes* saling berelasi [11].

Neo4j adalah salah satu *open-source*, NoSQL, dan *native graph database* yang menyediakan transaksi *backend* ACID untuk sebuah aplikasi [12]. Neo4j disebut sebagai *native graph database* karena efisien dalam pengimplementasi properti model *graph* pada level penyimpanan. Berikut ini adalah alasan menggunakan Neo4j sebagai *graph database* [13]: Neo4j memberikan kinerja yang konsisten, *real-time* untuk *multi-hop queries* pada *datasets* yang besar dan saling terhubung, *high availibity*, akses control yang *role-based*, *developer friendly*, dan *horizontal scalability*. Neo4j menggunakan bahasa Cypher sebagai bahasa *query*-nya [14]. Cypher adalah bahasa *query* deklaratif yang membuat sistem manajemen *graph database* dapat dimengerti dan bekerja untuk segala jenis pengguna *database* [14].

MongoDB adalah salah satu *document-oriented database* NoSQL yang digunakan untuk penyimpanan dengan volume besar [15]. MongoDB memiliki beberapa keunggulan sebagai *document-oriented database* NoSQL yaitu mempunyai *high performance*, *high availability*, *horizontal scalability*, skema dinamis yang mendukung polimorfisme, dan mengurangi pengguna *joins* [16]. MongoDB menyimpan datanya sebagai JSON (Javascript Object Notation) *document* dalam representasi biner bernama BSON(Binary JSON) [17].

PostgreSQL adalah salah satu *open source object-relational database system* yang menggunakan dan memperluas bahasa SQL yang digabungkan dengan banyak fitur yang aman dalam penyimpanan dan mengskala *data workloads* paling rumit [18]. PostgreSQL dikenal baik akan arsitekturnya, reliabilitasnya, *data integrity*, *robust*, ekstensibilitas, dan dedikasinya akan komunitas *open source*.

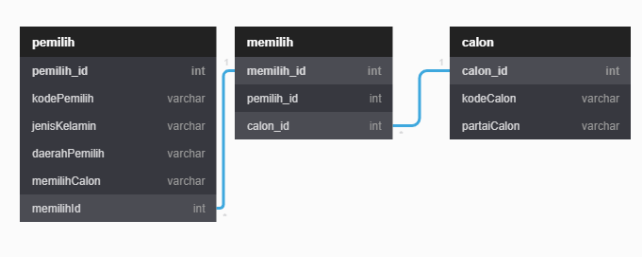
3. Sistem yang Dibangun

Pada penelitian terdapat 5 proses yang dilakukan dalam penelitian ini antara lain, *generate dataset*, *database modelling*, penerap *query*, uji performansi dari ketiga *database*, dan yang terakhir adalah menganalisis hasil dari uji performansinya menggunakan *Big O Notation*.

Tahap pertama adalah *generate dataset*. Pada tahap ini penulis membuatan *dataset* yang digunakan pada pengujian *database*. Data yang digunakan merupakan data simulasi yang bersumber dari data hasil rekapitulasi Pemilu DPR RI 2019 Dapil Jawa Barat pada website Komisi Pemilihan Umum (KPU) [19]. Lalu data hasil rekapitulasi tersebut penulis *generate* menggunakan *source code* dari *script python* untuk menunjukkan performansi jumlah data yang lebih besar. Sehingga diperoleh dua jenis data yaitu, pemilih dan calon. Jumlah total data pemilih ada 33,270,844 pemilih. Lalu untuk jumlah total data calon adalah 130 calon. Total keseluruhan data adalah 33,270,974 data. Pada data pemilih terdapat kolom pemilih_id, kodePemilih, daerahPemilih, jenisKelamin, memilihCalon dan pada data calon terdapat kolom calon_id, kodeCalon, partaiCalon.

Lalu tahap yang kedua adalah *database modelling*. Pada tahap ini penulis membuat skema untuk ketiga *database* yang penulis gunakan pada penelitian ini. Untuk model *database* pada PostgreSQL penulis menggunakan skema relasi. MongoDB menggunakan skema dokumen dengan format JSON. Sedangkan Neo4j menggunakan skema *graph.*

Adapun bentuk dari skema relasi *database* PostgreSQL adalah sebagai berikut.



Gambar 1. Skema relasi PostgreSQL

Pada gambar 1 diatas terdapat 3 tabel yang digunakan, yaitu tabel pemilih, tabel memilih, dan tabel calon. Tabel pemilih merupakan tabel entitas pemilih yang berisikan data pribadi pemilih. Dimana atribut pemilih_id pada tabel ini berperan sebagai *primary-key*. Lalu tabel memilih merupakan tabel relasi yang menghubungkan (JOIN) pemilih dengan calon yang akan dipilih. Atribut memilih_id pada tabel ini berperan sebagai *primary-key* sedangkan atribut pemilih_id berperan sebagai *foreign-key* dari tabel pemilih dan atribut calon_id berperan sebagai *foreign-key* dari tabel calon.Yang terakhir tabel calon merupakan tabel entitas calon yang berisikan data pribadi calon. Atribut calon_id pada tabel calon berperan sebagai *primary-key*.

Lalu untuk skema *document* MongoDB adalah sebagai berikut.

Gambar 2. Skema document pada MongoDB

Pada gambar 2 diatas hanya ada 1 *collection* yang digunakan, yaitu pemilu. Skema dokumen ini mengimplementasikan pola model *one-to-many relationship with embedded documents*. Dimana setiap dokumen pada *collection* pemilu memiliki *embedded sub-document* bernama hasilVoting pada field memilih. Dokumen utama *collection* pemilu terdiri dari field id, idPemilih, kodePemilih, daerahPemilih, jenisKelamin dan memilih. Lalu untuk *Embedded sub-document* hasilVoting memiliki field idCalon, kodeCalon, dan partaiCalon.

Terakhir skema *graph* untuk Neo4j adalah sebagai berikut.

Gambar 3. *Graph model* pada Neo4j

Pada *graph model* gambar diatas terdapat dua *nodes* yang digunakan, yaitu *nodes* pemilih dan *nodes* calon. *Nodes* pemilih berisikan atribut pemilih yaitu idPemilih, kodePemilih, daerahPemilih, jenisKelamin, dan memilih. Lalu pada *nodes* calon berisikan atribut calon yaitu idCalon, nomorCalon, kodeCalon, dan partaiCalon. *Nodes* pemilih dan *nodes* calon dihubungkan oleh *relationship* yang bernama memilih. Definisi *relationship* memilih adalah pemilih menggunakan hak pilihnya untuk memilih satu calon. Dapat dilihat dari ketiga model tersebut PostgreSQL, MongoDB, dan Neo4j memiliki perbedaan dalam skema *database*nya. Pada skema PostgreSQL. Untuk menghubungkan dua tabel harus ada sebuah table penghubung untuk menghubungkan kedua tabel tersebut dan *foreign-key*. Berbeda dengan MongoDB dan Neo4j yang tidak perlu ada tabel penghubung dan *foreign-key*. Karena keduanya memiliki sifat fleksibel dalam skema *database*nya [20]. Sifat fleksibel ini membuat MongoDB dan Neo4j tidak perlu menyimpan data pada tabel yang sama. Sehingga cukup menghubungkan lansgung kedua data tersebut. Sifat fleksibel MongoDB dan Neo4j yang lain adalah tidak perlu menetapkan skemanya di awal dan bisa mengubah skemanya bila ada perubahan pada *business requirements*.

Selanjutnya pada tahap membangun *database*. Tahap ini adalah tahap membangun *database* yang sesuai dengan model *database* yang telah dibuat pada tahap sebelumnya. Ada tiga *database* yang telah penulis bangun, yaitu Neo4j untuk *graph database*, MongoDB untuk *document-based database*, dan PostgreSQL untuk *relational database*.

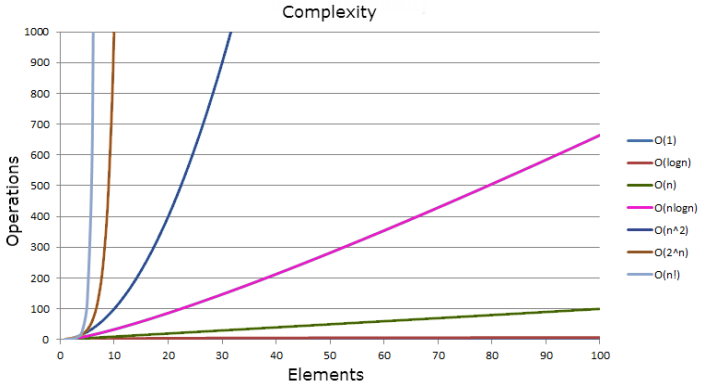
Setelah itu tahap penerapan *query*. Pada tahap ini adalah menerapkan *query* yang akan digunakan pada uji performansi ketiga *database*. *Query* yang digunakan pada penelitian ini adalah *query searching* dengan batas data yang dicari adalah 1,000,000 data dan tiap *dataset* memiliki *relationship*. Berikut ini adalah *query* yang akan diterapkan:

- *Query* 1: Mencari data pemilih yang berjenis kelamin perempuan yang memilih partai A
- *Query* 2: Mencari data pemilih yang berada di daerah Bogor yang memilih Calon A-3

Dan yang terakhir adalah analisis hasil uji performansi. Pada tahap ini penulis menggunakan *Big O Notation* untuk mengetahui performa dari ketiga *database*. *Big O Notation* adalah sebuah notasi yang menjelaskan kompleksitas dari algoritma menggunakan pendekatan aljabar [21]. Kompleksitas dari algoritma ini diukur berdasarkan durasi waktu algoritma itu memproses dengan jumlah inputan data sebanyak (n). *Big O Notation* telah diklasifikasikan ke beberapa notasi yaitu [22]:

- *Constant* O(1): Waktu eksekusi algoritma selalu sama, berapapun jumlah datanya.
- *Logarithmic* O(log n) Waktu eksekusi tumbuh secara logaritmik sejajar dengan n.
- *Linear* O(n) Waktu eksekusi tumbuh secara langsung sebanding dengan n.
- *A superlinear algorithm* – O(n log n): Waktu eksekusi tumbuh secara sebanding dengan n.
- *A polynomial algorithm* – O(n^c): Tumbuh lebih cepat dari sebelumnya berdasarkan n.
- *A exponential algorithm* – O(c^n): Waktu eksekusi tumbuh lebih cepat dari algoritma *polynomial*.
- *A factorial algorithm* – O(n!)Waktu eksekusi tumbuh secara drastis dan tidak dapat dipakai bahkan pada nilai n terkecil sekalipun.

Untuk mengetahui baik atau buruknya performa *database* berdasarkan notasi *Big* O. Dapat dilihat pada grafik dibawah ini [23].



Gambar 4. Grafik *Big O Notation*

Berdasarkan grafik diatas. Sebuah algoritma disebut paling bagus apalagi bila mempunya notasi O(1). Disebut lebih bagus apabila mempunyai notasi O(log n). Sebuah algoritma disebut bagus apabila mempunyai notasi O(n). Disebut buruk apabila mempunya notasi O($n \log n$). Dan disebut paling buruk apabila mempunyai notasi O($n!$), O(c^n), dan O(n^c). Pada penelitian ini ketiga *database* ini diuji berdasarkan durasi waktu eksekusi setiap *query*. Setelah selesai di-*query*, hasil waktu eksekusi query tersebut ditampilkan dalam bentuk grafik. Lalu untuk membuktikan kompleksitas *database*-nya, grafik waktu eksekusi *query* pada ketiga *database* dibandingkan dengan grafik *Big O Notation.*

4. Evaluasi

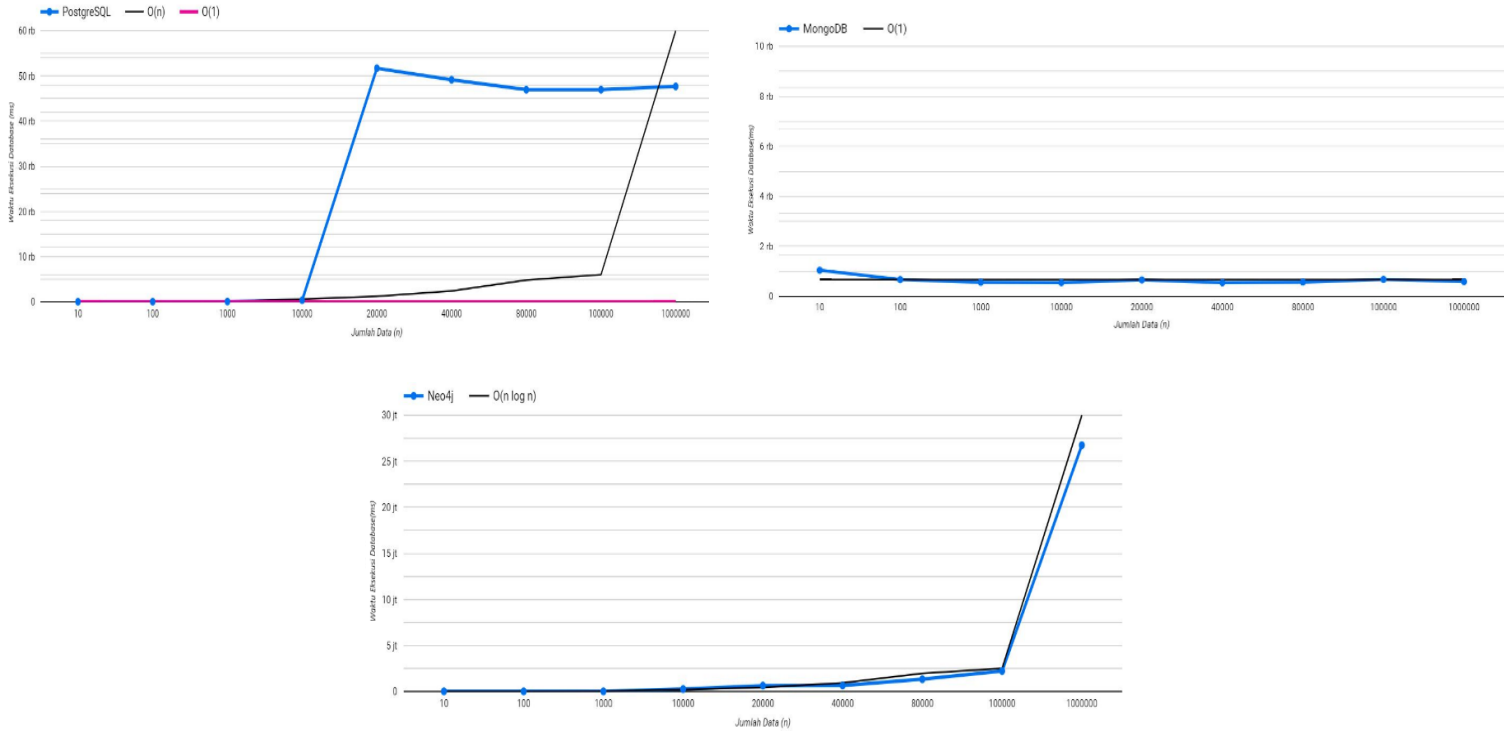
Hasil Pengujian *Query* 1 Mencari Data Pemilih yang Berjenis Kelamin Perempuan yang Memilih Partai A

Hasil dari pengujian *Query* 1 mencari data pemilih yang berjenis kelamin perempuan yang memilih Partai A adalah sebagai berikut.

Tabel 1. Hasil pengujian Query 1 pada PostgreSQL, MongoDB, dan Neo4j beserta *Big O Notation* pembanding (waktu dalam ms)

	Jumlah Data (n)...	PostgreSQL	MongoDB	Neo4j	O(1)	O(log n)	O(n)	O(n log n)	O(n^2)	O(n^3)
1.	10	6	1.040	30	1	1	10	10	100	1.000
2.	100	7	665	34	1	2	100	200	10.000	1.000.000
3.	1000	44	559	149	1	3	1.000	3.000	1.000.000	1.000.000.000
4.	10000	330	547	246.844	1	4	10.000	40.000	100.000.000	1.000.000.000.000
5.	20000	51.707	652	622.786	1	4	20.000	86.021	400.000.000	8.000.000.000.000
6.	40000	49.151	543	641.592	1	5	40.000	184.082	1.600.000.000	64.000.000.000.000
7.	80000	46.950	566	1.318.239	1	5	80.000	392.247	6.400.000.000	512.000.000.000.000
8.	100000	46.956	672	2.213.157	1	5	100.000	500.000	10.000.000.000	1.000.000.000.000.0...
9.	1000000	47.684	587	26.752.505	1	6	1.000.000	6.000.000	10.000.000.000.000.000....	100.000.000.000.000...

Pada kolom waktu eksekusi PostgreSQL untuk *range* data $10 < n < 20{,}000$ memiliki pola gradasi warna yang sama dengan kolom notasi O(n). Kemudian untuk *range* data $20000 < n < 1000000$ memiliki pola gradasi warna yang sama dengan kolom notasi O(1). Lalu pada kolom waktu eksekusi MongoDB untuk range data $10 < n < 1{,}000{,}000$ memiliki pola gradasi warna yang sama dengan kolom notasi O(1). Terakhir pada kolom waktu eksekusi Neo4j untuk *range* data $10 < n < 1{,}000{,}000$ memiliki pola gradasi warna yang sama dengan kolom notasi O($n \log n$). Lalu untuk melihat pertumbuhan hasil waktu eksekusi ketiga *database* yang lebih detailnya dapat dilihat pada gambar 5 sebagai berikut.



Gambar 5. Grafik hasil pengujian Query 1 pada PostgreSQL, MongoDB, dan Neo4j beserta *Big O Notation* pembanding (waktu dalam ms)

Berdasarkan gambar grafik diatas, pada grafik waktu eksekusi PostgreSQL untuk range data $10 < n < 20{,}000$ mendekati pertumbuhan yang sama dengan grafik O(n). Lalu untuk *range* data $20000 < n < 1000000$ mendekati pertumbuhan yang sama dengan grafik O(1). Sehingga pada pengujian *Query* 1 untuk PostgreSQL memiliki dua kompleksitas. Yaitu O(n) dan O(1). Lalu pada grafik waktu eksekusi MongoDB untuk range data $10 < n < 1{,}000{,}000$ mendekati pertumbuhan yang sama dengan grafik O(1). Sehingga pada pengujian *Query* 1 pada MongoDB memiliki kompleksitas O(1). Terakhir pada grafik waktu eksekusi Neo4j untuk *range* data $10 < n < 1{,}000{,}000$ mendekati pertumbuhan yang sama dengan grafik O($n \log n$). Sehingga pada pengujian *Query* 1 untuk Neo4j memiliki kompleksitas O($n \log n$).

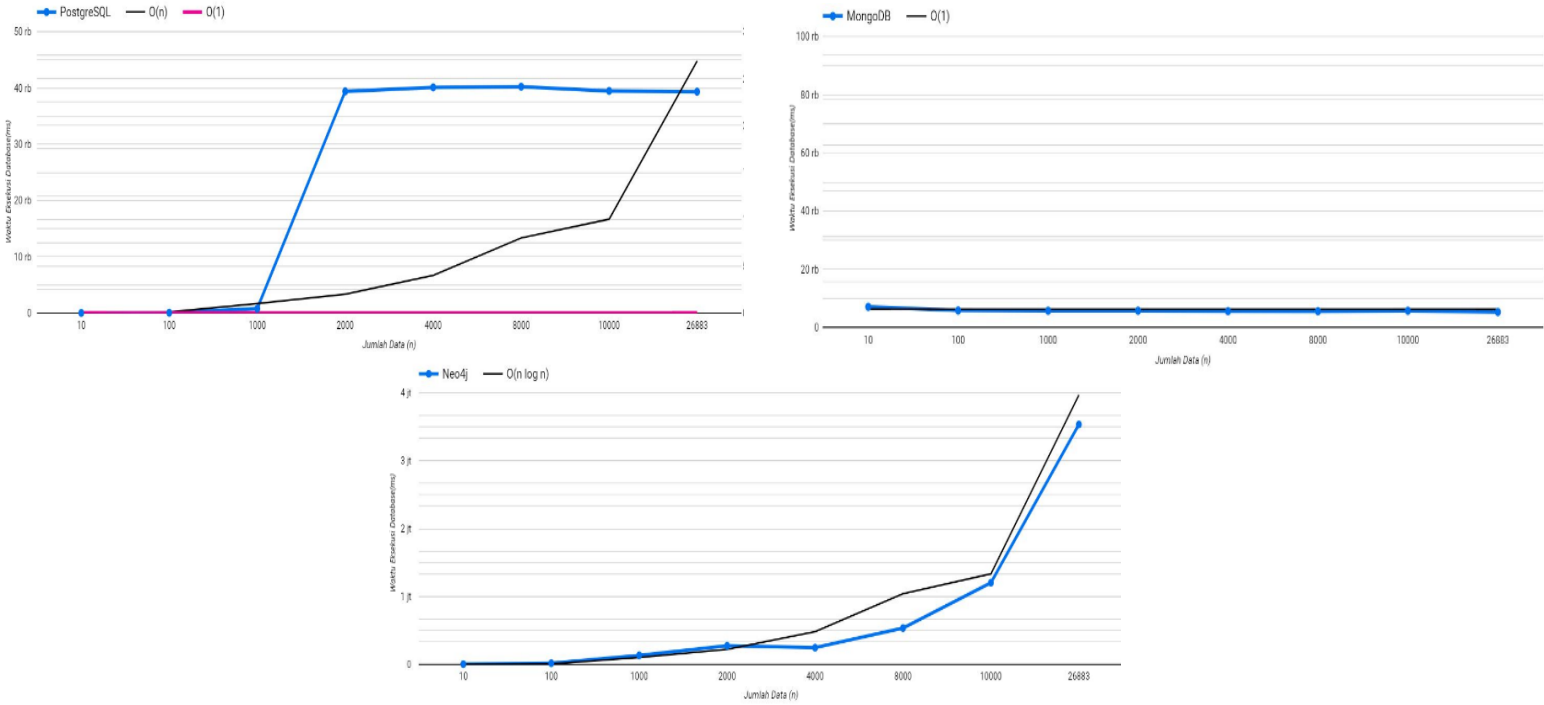
Hasil Pengujian *Query* 2 Mencari data pemilih yang berada di daerah Bogor yang memilih Calon A-3

Hasil dari pengujian *Query* 2 mencari data pemilih yang berada di daerah Bogor yang memilih calon A-3 adalah sebagai berikut.

Tabel 2. Hasil pengujian Query 2 pada PostgreSQL, MongoDB, dan Neo4j beserta *Big O Notation* pembanding (waktu dalam ms)

	Jumlah Data (n)	PostgreSQL	MongoDB	Neo4j	O(1)	O(log n)	O(n)	O(n log n)	O(n^2)	O(n^3)
1.	10	6	7.060	2.017	1	1	10	10	100	1.000
2.	100	43	5.800	16.043	1	2	100	200	10.000	1.000.000
3.	1000	771	5.710	131.669	1	3	1.000	3.000	1.000.000	1.000.000.000
4.	2000	39.399	5.720	273.018	1	3	2.000	6.602	4.000.000	8.000.000.000
5.	4000	40.101	5.550	244.633	1	4	4.000	14.408	16.000.000	16.000.000.000
6.	8000	40.225	5.530	533.541	1	4	8.000	31.225	64.000.000	512.000.000.000
7.	10000	39.465	5.700	1.202.213	1	4	10.000	40.000	100.000.000	1.000.000.000.000
8.	26883	39.340	5.270	3.534.872	1	4	26.883	119.078	722.695.689	19.428.228.207.387

Pada kolom waktu eksekusi PostgreSQL untuk *range* data $10 < n < 2{,}000$ memiliki pola gradasi warna yang sama dengan kolom notasi O(n). Kemudian untuk *range* data $2{,}000 < n < 26{,}833$ memiliki pola gradasi warna yang sama dengan kolom notasi O(1). Lalu pada kolom waktu eksekusi MongoDB untuk range data $10 < n < 26{,}883$ memiliki pola gradasi warna yang sama dengan kolom notasi O(1). Terakhir pada kolom waktu eksekusi Neo4j untuk *range* data $10 < n < 26{,}883$ memiliki pola gradasi warna yang sama dengan kolom notasi O($n \log n$). Lalu untuk melihat pertumbuhan hasil waktu eksekusi ketiga *database* yang lebih detailnya dapat dilihat pada gambar 6 sebagai berikut.



Gambar 6. Grafik hasil pengujian Query 2 pada PostgreSQL, MongoDB, dan Neo4j beserta *Big O Notation* pembanding (waktu dalam ms)

Berdasarkan gambar grafik diatas, pada grafik waktu eksekusi PostgreSQL untuk range data $10 < n < 2{,}000$ mendekati pertumbuhan yang sama dengan grafik O(n). Lalu untuk *range* data $2{,}000 < n < 26{,}883$ mendekati pertumbuhan yang sama dengan grafik O(1). Sehingga pada pengujian *Query* 2 untuk PostgreSQL memiliki dua kompleksitas. Yaitu O(n) dan O(1). Lalu pada grafik waktu eksekusi MongoDB untuk range data $10 < n < 26{,}883$ mendekati pertumbuhan yang sama dengan grafik O(1). Sehingga pada pengujian *Query* 2 pada MongoDB memiliki kompleksitas O(1). Terakhir pada grafik waktu eksekusi Neo4j untuk *range* data $10 < n < 26{,}883$ mendekati pertumbuhan yang sama dengan grafik O($n \log n$). Sehingga pada pengujian *Query* 2 untuk Neo4j memiliki kompleksitas O($n \log n$).

Analisis Hasil Pengujian

Berdasarkan hasil uji dari keempat *query*, MongoDB memiliki performa yang sangat bagus untuk digunakan sebagai *database* manajemen *big data* dengan kompleksitas O(1). Hal ini terjadi karena MongoDB menggunakan metode pencarian COLLSCAN atau *collection scan* dalam *query* pencariannya yang melakukan pencarian pada satu *collection*. Kemudian tipe *database document-based* seperti MongoDB ini memudahkan penyimpanan dan query datanya ke dalam satu *collection* yang sama. Sehingga tidak perlu ada *collection* tambahan untuk menghubungkan antar *collection* (JOIN). Karena hubungan antara datanya sudah didefiniskan pada *collection* yang sama.

Lalu pada PostgreSQL memiliki performa yang baik dengan gabungan kompleksitas O(1) dan O(n). Hal ini terjadi klausa SELECT pada PostgreSQL melakukan "*full table scan*" yang menggunakan algoritma *sequential search*. *Sequential search* pada PostgreSQL memeriksa satu persatu data yang ada di table sampai data yang dicari telah ditemukan. Pada penelitian ini, proses *sequential search* dilakukan secara paralel karena terdapat 3 tabel yang digunakan. Setelah semua datanya ditemukan, PostgreSQL melakukan JOIN pada ketiga tabel tersebut kemudian melakukan *filtering* untuk menampilkan data yang dicari.

Terakhir pada Neo4j memiliki performa yang buruk dengan kompleksitas O($n \log n$). Hal ini terjadi karena *platform* cypher pada Neo4j sangat lambat untuk meng-*query* data yang jumlahnya besar. Dimana pada penelitian ada 33,270,974 data. Cypher memeriksa keseluruhan *nodes* pada *database* yang memiliki *property* yang sama dengan kriteria pada *query* di cypher tersebut. Lalu *nodes* yang sudah sama akan di-filter untuk memberikan hasil *query* tersebut. Proses inilah yang membuat Neo4j memiliki waktu *query* yang sangat lama ketika melakukan *query* terhadap data yang berjumlah 1,000,000.

5. Kesimpulan

Dari penelitian ini, kesimpulan yang bisa diambil adalah MongoDB memiliki performa yang paling bagus digunakan sebagai *database* manajemen *big data* pemilu dengan kompleksitas O(1). Selain itu MongoDB juga memiliki skema yang fleksibel. PostgreSQL memiliki performa yang baik dengan gabungan kompleksitas O(1) dan O(n). Neo4j memiliki performa yang buruk dengan kompleksitas O($n \log n$). Namun Neo4j masih bisa digunakan sebagai *database* alternatif karena skemanya yang fleksibel. Saran dari penulis untuk penelitian kedepannya adalah untuk melakukan pengujian performa pada PostgreSQL, MongoDB, dan Neo4j dengan

faktor jumlah memori yang ketiga *database* tersebut gunakan. Karena ada kemungkinan faktor penggunaan memori ini mempengaruhi performa ketiga *database* tersebut dalam melakukan *query*.

Daftar Pustaka

[1] C. G. Asmara, "Jokowi: Data Lebih Berharga dari Minyak," CNBC Indonesia, 16 August 2019. [Online]. Available: https://www.cnbcindonesia.com/tech/20190816112736-37-92469/jokowi-data-lebih-berharga-dari-minyak. [Accessed 11 October 2019].

[2] Iskandar, "Data Jadi Komoditas Utama di Era Digital," Liputan6, 11 September 2019. [Online]. Available: https://www.liputan6.com/tekno/read/4059254/data-jadi-komoditas-utama-di-era-digital. [Accessed 18 October 2019].

[3] Y. Permana, "Mengenal Big Data," Codepolitan, 29 May 2016. [Online]. Available: https://www.codepolitan.com/mengenal-big-data. [Accessed 2019 October 11].

[4] Y. Debora, "Big Data: Kunci Pemenangan Pilkada," tirto.id, 19 December 2017. [Online]. Available: https://tirto.id/big-data-kunci-pemenangan-pilkada-cBXY. [Accessed 11 October 2019].

[5] I. Robinson, J. Webber and E. Eifrem, Graph *Database*s, United States of America: O'Reilly Media, Inc, 2015.

[6] M. Sharma, V. D. Sharma and M. M. Bundele, "Performance Analysis of RDBMS and No SQL *Database*s : PostgreSQL , MongoDB and Neo4j," in *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, 2018.

[7] I. N. D. Pande and R. Sarno, "Book Recommendation Using Neo4j Graph *Database* in BibTeX Book Metadata," in *2017 3rd International Conference on Science in Information Technology (ICSITech)*, Bandung, 2017.

[8] H. Lu, Z. Hong and M. Shi, "Analysis of Film Data Based on Neo4j," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 2017.

[9] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen and D. Wilkins, "A Comparison of a Graph *Database* and a Relational *Database*: A Data Provenance Perspective," in *Proceedings of the 48th Annual Southeast Regional Conference*, Oxford, 2010.

[10] J. Hurwitz, N. Alan, F. Halper and M. Kaufman, Big Data For Dummies, Hoboken: John Wiley & Sons, Inc., 2013.

[11] B. M. Sasaki, J. Chao and R. Howard, Graph *Database*s For Beginners, Neo4j, 2018.

[12] "What is a Graph *Database*?," Neo4j, [Online]. Available: https://neo4j.com/developer/graph-*database*/. [Accessed 30 October 2019].

[13] "The Native Graph *Database* for Today's Connected Applications," Neo4j, [Online]. Available: https://neo4j.com/neo4j-graph-*database*/. [Accessed 31 October 2019].

[14] R. V. Bruggen, Learning Neo4j, Birmingham: Packt Publishing, 2014.

[15] "What is MongoDB? Introduction, Architecture, Features & Example," Guru99, [Online]. Available: https://www.guru99.com/what-is-mongodb.html#1. [Accessed 29 October 2019].

[16] "Introduction to MongoDB," MongoDB, [Online]. Available: https://docs.mongodb.com/manual/introduction/#document-*database*. [Accessed 29 October 2019].

[17] MongoDB Architecture Guide: Overview, MongoDB, 2019.

[18] "About," PostgreSQL, [Online]. Available: https://www.postgresql.org/about/. [Accessed 30 October 2019].

[19] "Lindungi Hak Pilihmu," Komisi Pemilihan Umum, [Online]. Available: https://lindungihakpilihmu.kpu.go.id/. [Accessed 28 Februari 2020].

[20] K. Sahatqija, J. Ajdari, X. Zenuni, B. Raufi and F. Ismaili, "Comparison between relational and NOSQL *database*s," in *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, 2018.

[21] S. Huang, "What is *Big O Notation* Explained: Space and Time Complexity," freeCodeCamp, 16 Januari 2020. [Online]. Available: https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt-1674cfa8a23c/. [Accessed 22 Juni 2020].

[22] S. Debnath, "Analysis of Algorithms | Big-O analysis," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/. [Accessed 27 June 2020].

[23] D. Newton, "Learning *Big O Notation* With O(n) Complexity," DZone, 25 April 2017. [Online]. Available: https://dzone.com/articles/learning-big-o-notation-with-on-complexity. [Accessed 7 Juli 2020].