Jurnal ELTIKOM :
Jurnal Teknik Elektro, Teknologi Informasi dan Komputer

Vol. 6, No. 1, Juni 2022, hal. 89-99
ISSN 2598-3245 (Print), ISSN 2598-3288 (Online)
DOI: http://doi.org/10.31961/eltikom.v6i1.532
Tersedia online di http://eltikom.poliban.ac.id

PENGUJIAN OPTIMIZATION DAN NON-OPTIMIZATION QUERY METODE TOPSIS UNTUK MENENTUKAN TINGKAT KERUSAKAN SEKTOR BENCANA ALAM

Annisa Heparyanti Safitri, Agung Teguh Wibowo Almais*, A'la Syauqi , Roro Inda Melani
Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Malang, Indonesia
e-mail: {annisahepar, agung.twa}@gmail.com, {syauqi, roro.melani}@ti.uin-malang.ac.id

Diterima: 19 September 2021 – Direvisi: 29 November 2021 – Disetujui: 29 November 2021

ABSTRACT

The huge volume of data from the Disaster Management Planning and Control (P3B) surveyor team creates wide and varied problems that can consume system resources and processing time that is relatively long. Therefore, this study proposes a solution by performing query optimization on the TOPSIS method which is implemented in a decision support system to determine the level of post-disaster damage. Based on 3 trials with different amounts of data, the 1st trial used 114 data, the 2nd trial used 228 data and the 3rd trial used 334 data. In addition, for each trial, the response time measurement was repeated 3 times, so that the average response time of each step of the TOPSIS method was obtained. It was found that the results of the ranking stage using query optimization were 0.00076 faster than the non-optimization query. So, it can be concluded that the response time obtained by query optimization at each step of the TOPSIS method in the post-natural disaster sector damage decision support system is smaller than the response time in non-optimization queries.

Keywords*: Query Optimization, Response time, TOPSIS.*

ABSTRAK

Volume data yang sangat besar dari tim surveyor Perencanaan dan Pengendalian Penanganan Bencana(P3B) menciptakan masalah yang luas dan beragam sehingga dapat menghabiskan sumber daya sistem dan waktu pemrosesan yang terbilang lama. Oleh karena itu penelitian ini mengusulkan solusi dengan melakukan Optimasi query pada metode TOPSIS yang diimplementasikan pada sistem pendukung kepeutusan untuk menentukan tingkat kerusakan pasca bencana. Berdasarkan 3 kali uji coba dengan jumlah data yang berbeda-beda yaitu ujicoba ke-1 menggunakan 114 data, ujicoba ke-2 sebanyak 228 data dan ujicoba ke-3 menggunakan 334 data. Selain itu, setiap ujicoba dilakukan lagi pengukuran respons time sebanyak 3 kali maka didapatkan hasil rata-rata (average) response time dari masing-masing langkah metode TOPSIS. Didapati bahwa hasil dari tahapan perangkingan menggunakan query optimization lebih cepat 0.00076 dibandingakan dengan qury non-optimization. Sehingga dapat di simpulkan bahwa response time yang didapat query optimization pada setiap langkah metode TOPSIS pada sistem pendukung keputusan kerusakan sektor pasca bencana alam lebih kecil dibandingkan dengan response time pada query non-optimization.

Kata Kunci: *Query Optimization, Response time, TOPSIS.*

I. PENDAHULUAN

BENCANA alam yang sering terjadi di Indonesia diantaranya yaitu banjir, kebakaran, puting beliung, dan tanah longsor. Sebagian besar setiap provinsi di Indonesia mempunyai catatan bencana alam yang telah terjadi. Dampak dari kejadian bencana alam tersebut sering sekali menimbulkan kerugian berupa harta benda dan korban jiwa [1]. Sekitar 13% dari gunung berapi dunia yang bertempat di kepulauan Indonesia memiliki potensi untuk memicu bencana dengan kekuatan dan intensitas yang berbeda. Penanggulangan bencana alam merupakan upaya selanjutnya untuk mengurangi dampak dari bencana alam terhadap manusia dan harta benda [2].

Instansi pemerintah yang memiliki tanggung jawab untuk menangani terkait penanggulangan bencana yaitu Badan Nasional Penanggulangan Bencana atau dikenal juga dengan BNPB yang merupakan wadah yang bersifat non struktural bagi penanggulangan bencana yang berada di bawah Presiden dan bertanggungjawab langsung kepada Presiden. Penanggulangan yang ditangani oleh BPBD berupa pra maupun pasca bencana alam [3]. Persiapan dalam menghadapi bencana alam sendiri meliputi aktivitas yang dilakukan sebelum terdeteksinya tanda-tanda bencana sehingga bisa memfasilitasi pemakaian sumber daya yang tersedia, meminta bantuan serta merencanakan rehabilitasi dalam cara dan kemungkinan yang paling baik [2].

Keberhasilan Tim Perencanaan dan Pengendalian Penanganan Bencana (P3B) dalam program pemulihan pascabencana sangat ditentukan oleh perencanaan pemulihan yang baik dari data dan informasi yang akurat. Banyaknya permasalahan di lapangan disebabkan oleh ketidakakuratan data. Hal ini terjadi karena kriteria pengumpulan data awal yang digunakan oleh surveyor di lapangan dipersepsikan berbeda sehingga kategorisasi data yang ada menjadi berbeda. Dengan demikian, timbul masalah dan data bencana yang masuk ke BPBD Provinsi berbeda dengan keadaan di lapangan karena kriteria data untuk persiapan tindakan rehabilitasi dan rekonstruksi pascabencana yang digunakan oleh masing-masing surveyor dianggap berbeda [4].

Dari identifikasi permasalahan di atas, diperlukan suatu sistem untuk membantu tim surveyor dalam proses persiapan tindakan rehabilitasi dan rekonstruksi pasca bencana. Penggunaan sistem yang dilengkapi dengan *Decision Support System* (DSS) diharapkan dapat mempercepat tim surveyor dalam memproses tindakan rehabilitasi dan rekonstruksi pasca bencana alam, serta dapat meningkatkan akurasi dalam memperoleh hasil yang tepat [5]. Namun, volume data yang sangat besar ini menciptakan masalah yang tidak terselesaikan untuk eksekusi query DSS. Query DSS yang luas dan beragam dapat menghabiskan sumber daya sistem dan waktu pemrosesan yang terbilang lama. Untuk mengoptimalkan sumber daya dan untuk mempercepat pengambilan data klinis dan proses analisis, query DSS harus dioptimalkan secara efektif [6].

Pemrosesan query baru-baru ini muncul sebagai solusi yang layak untuk menangani data yang jumlahnya besar, kompleksitas query yang tinggi, yang menjadi ciri aplikasi DSS saat ini. Biasanya, pengguna DSS memiliki sikap yang sangat kompleks untuk query *Database Manajemen System* (DBMS) yang mendasari dan memerlukan operasi kompleks sampai *Gigabyte* atau *Terabyte* dari ruang penyimpanan. Dengan membutuhkan waktu yang sangat lama untuk dieksekusi hingga selesai agar menghasilkan jawaban yang tepat [7].

Oleh karena itu penelitian ini mengusulkan solusi dengan melakukan Optimasi query pada metode TOPSIS untuk diterapkan dalam DSS. Optimasi query adalah tugas yang dapat menjadikan trigger dari setiap sistem database. Sejumlah heuristik telah diterapkan belakangan ini, yang mengusulkan algoritma baru untuk secara substansial meningkatkan kinerja query. Perkembangan yang tidak dapat hilang dalam bidang database DSS adalah menyajikan data dengan kecepatan yang luar biasa [8]. Selain itu, penelitian yang dilakukan oleh Kuo [9] mengungkapkan bahwa sebagai alat analisis keputusan, TOPSIS (*Technique for Order Preference by Similarity to Ideal Solution*) berupaya untuk memilih alternatif yang secara simultan memiliki jarak terdekat dari *Solusi Ideal Positif* (PIS) dan jarak terjauh dari *Solusi Ideal Negatif* (NIS). Optimasi query dalam sistem database telah mendapatkan perhatian yang cukup besar dalam beberapa tahun terakhir begitu pula dengan penggunaan metode TOPSIS sebagai metode TOPSIS. Proses untuk menentukan eksekusi query terbaik dapat menggunakan dengan *response time*.

Response time yaitu waktu tanggap yang diberikan oleh interface ketika user mengirim request ke komputer. Secara umum, user menginginkan bahwa program aplikasinya bisa memberikan waktu tanggap yang sesingkat-singkatnya. Akan tetapi waktu tanggap yang ideal tidak dapat ditentukan, karena ada beberapa aspek yang mempengaruhi, seperti ragam interaksi yang diinginkan dan kelancaran pengguna dalam menjalankan program aplikasi tersebut [10].

Volume data yang sangat besar dari tim surveyor Perencanaan dan Pengendalian Penanganan Bencana (P3B) menciptakan masalah yang luas dan beragam sehingga dapat menghabiskan sumber daya sistem dan waktu pemrosesan yang terbilang lama. Maka diharapakan dengan penerapan *query optimization* terhadapa query TOPSIS akan mempermudah tim surveyor baik dari segi efesiensi waktu dan efesiensi *storage* data bencana yang masuk ke BPBD Provinsi.

TABEL 1
LANGKAH-LANGKAH METODE TOPSIS

Langkah	Penjelasan	Persamaan	
1.	Rumuskan matriks keputusan ternormalisasi R	$R = (r_{ij})_{m\times n} = \left(\frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}\right)_{m\times n}$	(1)
2.	Mengembangkan matriks keputusan D	$D = (x_{ij})_{m\times n}$	(2)
3.	Membangun matriks keputusan ternormalisasi dengan bobot V	$V = (v_{ij})_{m\times n} = (w_j\, r_{ij})_{m\times n}$	(3)
4.	Mengidentifikasi solusi positif ideal (PIS) (+ = ... + + + A vvv (1 2 ,,, n)) dan solusi negatif ideal (NIS) (A- = (v1 -, v2 -, ... , vn -})	$v_j^+ = \{({}^{max}_{i} v_{ij} \mid j \in J_b), ({}^{max}_{i} v_{ij} \mid j \in J_c) \mid i \in [1..m]\}$	(4)
		$v_j^- = \{({}^{max}_{i} v_{ij} \mid j \in J_b), ({}^{max}_{i} v_{ij} \mid j \in J_c) \mid i \in [1..m]\}$	(5)
5.	Menghitung ukuran pemisahan (si + dan si -)	$s_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^+)^2} \;\forall\, i \in [1..m]$	(6)
		$s_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2} \;\forall\, i \in [1..m]$	(7)
6.	Cari koefisien kedekatan (CCi)	$CC_i = \frac{s_i^-}{(s_i^+ + s_i^-)} \forall i \in [1..m]$	(8)

II. METODE PENELITIAN

A. Data Collection

Data yang digunakan dalam penelitian ini berupa data sekunder, dimana data sekunder yaitu data yang didapatkan dari lembaga atau pihak peneliti lainnya. Data tersebut didapatkan dari rekapan bencana alam dari Badan Nasional Penanggulangan Bencana di Indonesia. Pada data tersebut terdapat beberapa kriteria yang dapat dijadikan acuan dalam penentuan kerusakan sektor pasca bencana. Dimana kriteria tersebut yang akan mendukung sistem yang akan dirancang untuk surveyor dalam menentukan kerusakan sektor pasca bencana. Data yang disajikan dibuat oleh ahli dalam bidang terkait, di dalamnya terdapat data bencana alam baik berupa kerusakan rumah, instansi, jembatan, maupun masjid.

B. TOPSIS

Pada penelitian [11] bahwa metode TOPSIS merupakan salah satu metode MCDM yang menentukan alternatifnya berdasarkan jarak terpendek dari nilai PIS dan nilai terpanjang dari nilai NIS. Secara umum metode TOPSIS menggunakan langkah-langkah pada Tabel 1. Berdasarkan rumus-rumus yang terdapat pada Tabel 1 dijelaskan sebagai berikut. Langkah pertama, identifikasi atribut esensial (parameter dependen dan independen) yang harus dilakukan secara primer. Umumnya, parameter dependen menginginkan maksimalisasi yang dianggap paling disukai dan parameter dependen menginginkan minimalisasi yang dianggap paling tidak disukai. Diketahui **rij** adalah nilai normalisasi matriks keputusan dan **xij** adalah nilai asli matriks keputusan. Untuk cara menghitung dapat dilihat pada Persamaan 1.

Selanjutnya langkah kedua terdapat pada Persamaan 2, matriks A lebih sering disebut sebagai matriks keputusan yang digunakan untuk mewakili semua informasi, yang memiliki **i** baris (m - alternatif) dan **j** kolom (n - kriteria).

Langkah ketiga, perkalian silang bobot individu dengan kolom masing-masing matriks keputusan ternormalisasi akan menghasilkan matriks keputusan ternormalisasi berbobot. Dimana **Wj** adalah bobot kriteria dan **Nij** adalah matriks yang dinormalisasi. Bobot (**Wj**) masing-masing kriteria ditentukan berdasarkan keahlian. Nilai normalisasi terbobot dihitung menggunakan Persamaan 3.

Kemudian untuk langkah keempat, estimasi solusi ideal positif dan solusi ideal negatif. Hal tersebut dianalisis dengan menggunakan Persamaan 4 dan 5. Diketahui bahwa J = 1, 2, 3,…, n dimana J berhubungan dengan kriteria manfaat J' = 1, 2, 3,…, n dimana J' berhubungan dengan kriteria biaya. Solusi ideal positif (**Vj+**) oleh Persamaan 4 dianggap berdasarkan minimalisasi sedangkan solusi ideal yang tidak menguntungkan (**Vj-**) oleh Persamaan 5 diperhitungkan berdasarkan maksimalisasi.

TABEL 2
BEBERAPA CARA UNTUK QUERY OPTIMASI

Perintah	Keterangan
Optimasi menggunakan index	Menggunkan index scan yaitu suatu cara memanggil suatu field yang ada pada suatu table, jadi tidak memanggil table yang bersangkutan.
Penentuan Tipe data	Dalam menentukan tipe data membutuhkan ketelitian dan analisa yang baik. Sebagai contoh kapan kita menggunkna tipe data char dan varchar.
Kurangi Penggunaan Allow Null	Sebagai gantinnya dapat menggunakan default. Nilai NULL dapat menambah beban query yang mengaksesnya.
Query yang Mudah Terbaca	Pemilihan huruf besar dan kecil dapat mempermudah pembacaan, misalnya dengan konsisten menuliskan keyword SQL dalam huruf kapital, dan tambahkan komentar bilamana diperlukan
Hindari SELECT *	SELECT * digunakan untuk melakukan query semua field yang terdapat pada sebuah table, tetapi jika hanya ingin memproses field tertentu, maka sebaiknya ditulis field yang ingin diakses saja, sehingga query menjadi SELECT field1, field2, field3 dan seterusnya.
Batasi ORDER BY	Penggunaan ORDER BY yang berfungsi untuk mengurutkan data, ternyata memiliki konsekuensi menambah beban query, karena akan menambah satu proses lagi, yaitu proses sort. Karena itu gunakan ORDER BY hanya jika benar-benar dibutuhkan oleh aplikasi.
Gunakan WHERE dalam SELECT	"di mana ada gula di sana ada semut". Untuk programer database, pepatah itu perlu dimodifikasi menjadi "di mana ada SELECT di sana ada WHERE", untuk mengingatkan pentingnya klausa WHERE sebagai kondisi untuk menyaring record sehingga meminimalkan beban jaringan.
Batasi Penggunaan Function	Gunakan fungsi-fungsi yang disediakan SQL seperlunya saja. Contohnya, SELECT nama FROM tbl_teman WHERE ucase(nama) = 'ABC', nampak query tersebut ingin mencari record yang memiliki data berisi "abc", fungsi ucase digunakan untuk mengubah isi field nama menjadi huruf besar dan dibandingkan dengan konstanta "ABC" untuk meyakinkan bahwa semua data "abc" akan tampil, walaupun dituliskan dengan huruf kecil, besar, ataupun kombinasinya.
Baca dari Kiri ke Kanan	Query yang ditulis akan diproses dari kiri ke kanan, misalkan terdapat query WHERE kondisi1 AND kondisi2 AND kondisi3, maka kondisi1 akan terlebih dahulu dievaluasi, lalu kemudian kondisi2, kondisi3, dan seterusnya. Tentunya dengan asumsi tidak ada kondisi yang diprioritaskan/dikelompokkan dengan menggunakan tanda kurung.
Gambar dalam Database	Gambar simpan link atau lokasi gambar di dalam database, dibandingkan menyimpan fisik gambar tersebut. Kecuali jika tidak memiliki pilihan lain, misalnya karena alasan keamanan atau tidak tersedianya tempat penyimpanan lain untuk gambar selain di dalam database.

TABEL 3
SPESIFIKASI *SOFTWARE* YANG DIGUNAKAN

No	Software	Version
1	MongoDB	10.4.11
2	DBVisualizer	12.0.8

TABEL 4
SPESIFIKASI *HARDWARE* YANG DIGUNAKAN

No	Item	Descriptions
1	Processor	AMD E2-7110 APU
2	Memory	8 GB
3	Operating System	Windows 10 Pro64-bit
4	Harddisk	500

Menganggap penelitian ini semua enam tanggapan harus diminimalkan untuk meningkatkan produktivitas dan kualitas komponen.

Untuk langkah kelima, diketahui bahwa ukuran separasi diperoleh dengan menggunakan Persamaan 6. Sama untuk setiap preferensi yang paling disukai ditentukan oleh Persamaan 6. Demikian pula yang tidak menguntungkan dihitung menggunakan Persamaan 7.

Terakhir, diketahui bahwa kedekatan relatif diperiksa menggunakan Persamaan 8 untuk setiap alternatif. Alternatif terbaik dipilih berdasarkan nilai CCi dan kedekatan dengan solusi akhir.

C. Query Optimization

MariaDB adalah salah satu database server yang digunakan untuk menyimpan dan memanajemen data. MariaDB bisa dikatakan mirp dengan MySQL, dikarenakan MariaDB adalah versi pengembangan terbuka dan mandiri dari MySQL [12]. MySQL mempunyai bagian berupa *Structured Query Language* (SQL) yang digunakan untuk mengolah database relasional yang ada didalamnya [13].

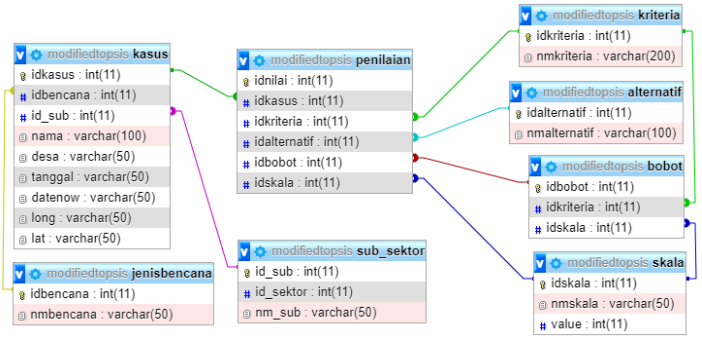
SQL mempunyai peran yang sangat penting guna meningkatkan kinerja suatu aplikasi atau sistem yang menggunakan database. Adapun salah satu cara untuk dapat melakukan hal tersebut maka perlu adanya optimasi suatu SQL. SQL dapat digunakan untuk mengakses database, mengambil data dari database (*retrieval*), menambahkan data kedatabase (*insert*), menghapus data didalam database (*delete*), dan mengubah data didalam database (*update*) [14]. Jika semua perintah SQL tersebut di optimalkan maka suatu sistem atau aplikasi akan meningkat kinerjanya.

TABEL 5
DATA ALTERNATIF

Alternatif	Nama Alternatif
A1	Rusak Ringan
A2	Rusak Sedang
A3	Rusak Berat

TABEL 6
KRITERIA

Kriteria	Nama Kriteria
C1	Kondisi Bangunan
C2	Kondisi Struktur Bangunan
C3	Kondisi Fisik Bangunan
C4	Fungsi Bangunan
C5	Kondisi Penunjang Lainnya



Gambar 1. Desain Database

Query optimization merupakan sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu query guna untuk membuat evaluasi sistem/ aplikasi/ metode menjadi lebih efektif [15]. Pada Tabel 2 adalah beberapa contoh atau cara yang digunakan untuk mengoptimalkan suatu query agar sistem/ aplikasi/ metode menjadi lebih efektif:

D. Identifikasi Kebutuhan Perangkat Lunak dan Perangkat Keras

Kebutuhan perangkat lunak dan keras dapat dilihat pada Tabel 3 dan 4.

E. Data

Data yang digunakan ujicoba menggunakan data kerusakan sektor paca bencana alam yang sudah di analisis, kemudian data tersebut di inputkan ke dalam metode TOPSIS dengan menggunakan sistem berbasis web. Data yang dimaksud adalah data kriteria, data alternatif, data pembobotan dan data penilaian alternatif berdasarkan kriteri.

Pada Tabel 5 menunjukan informasi alternatif yang didapatkan dari data Pusat Vulkanologi dan Mitigasi Bencana Geologi di Provinsi Jawa Timur. Dalam penelitian ini penulis mengambil data pola kerusakan bencana di kota Malang Provinsi Jawa Timur. Alternatif tersebut antara lain Rusak Ringan, Rusak Sedang dan Rusak Berat.

Pada Tabel 6 memberikan informasi Kriteria yang didapatkan dari data pola kerusakan bencana di Kota Blitar. Kriteria tersebut antara lain Kondisi Bangunan, Kondisi Struktur Bangunan, Kondisi Fisik Bangunan, Fungsi Bangunan dan Kondisi Penunjang Lainnya.

Setelah itu data disimpan pada database Maria DB untuk di proses menjadi metode TOPSIS. Untuk proses implementasi metode TOPSIS pada database MariaDB menggunakan *function view* yang ada pada library database (MariaDB). Setiap langkah yang terdapat pada Tabel 2 diatas di implementasikan dengan perintah query kemudian hasil dari query tersebut di simpan kedalam *view*. *View* merupakan suatu *function* yang berfungsi untuk menyimpan query, query tersebut berisi data-data yang akan digunakan pada langkah query metode TOPSIS selanjutnya. Hasil dari view hampir sama dengan table di database tetapi *view* tidak bisa melakukan CRUD (Create, Insert, Update, Delete) data yang tersimpan di view tersebut. Ada perubahan data yang di view jika ada perubahan data pada tabel yang bersangkutan dengan view tersebut. Untuk memanggil view dengan menggunakan perintah query ‘SELECT’ seperti pada saat memanggil data pada table di dalam database MariaDB.

TABEL 7
LANGKAH-LANGKAH METODE TOPSIS

TOPSIS Langkah ke-	Query Non-Optimization	Query Optimization
1	select `penilaian`.`idkasus` AS `idkasus`,`kriteria`.`idkriteria` AS `idkriteria`,`penilaian`.`idalternatif` AS `idalternatif`,sqrt(sum(pow(`skala`.`value`,2))) AS `bagi` from ((`penilaian` join `skala`) join `kriteria`) where `skala`.`idskala` = `penilaian`.`idskala` and `kriteria`.`idkriteria` = `penilaian`.`idkriteria` group by `kriteria`.`idkriteria`,`penilaian`.`idkasus`	SELECT `penilaian`.`idkasus` AS `idkasus`,`kriteria`.`idkriteria` AS `idkriteria`,`penilaian`.`idalternatif` AS `idalternatif`,SQRT(SUM(POW(`skala`.`value`,2))) AS `bagi` FROM ((penilaian JOIN skala) JOIN kriteria) WHERE `kriteria`.`idkriteria` = penilaian.idkriteria AND `skala`.`idskala` = penilaian.idskala GROUP BY `kriteria`.`idkriteria`,`penilaian`.`idkasus`
2	select `penilaian`.`idnilai` AS `idnilai`,`penilaian`.`idkasus` AS `idkasus`,`penilaian`.`idkriteria` AS `idkriteria`,`penilaian`.`idalternatif` AS `idalternatif`,`penilaian`.`idbobot` AS `idbobot`,`penilaian`.`idskala` AS `idskala`,`skala`.`value` / `1pembagi`.`bagi` AS `normalisasi` from ((`penilaian` join `1pembagi`) join `skala`) where `penilaian`.`idskala` = `skala`.`idskala` and `1pembagi`.`idkriteria` = `penilaian`.`idkriteria` group by `penilaian`.`idnilai`,`penilaian`.`idkasus`	SELECT `penilaian`.`idnilai` AS `idnilai`,`penilaian`.`idkasus` AS `idkasus`,`penilaian`.`idkriteria` AS `idkriteria`,`penilaian`.`idalternatif` AS `idalternatif`,`penilaian`.`idbobot` AS `idbobot`,`penilaian`.`idskala` AS `idskala`,`skala`.`value` / `1pembagi`.`bagi` AS `normalisasi` FROM ((`penilaian` JOIN `1pembagi`) JOIN `skala`) WHERE `1pembagi`.`idkriteria` = penilaian.idkriteria AND `penilaian`.`idskala` = skala.idskala GROUP BY `penilaian`.`idnilai`,`penilaian`.`idkasus`
3	select `2normalisasi`.`idnilai` AS `idnilai`,`2normalisasi`.`idkasus` AS `idkasus`,`2normalisasi`.`idkriteria` AS `idkriteria`,`2normalisasi`.`idalternatif` AS `idalternatif`,`2normalisasi`.`idbobot` AS `idbobot`,`2normalisasi`.`idskala` AS `idskala`,`2normalisasi`.`normalisasi` AS `normalisasi`,`skala`.`value` AS `value`,`skala`.`value` * `2normalisasi`.`normalisasi` AS `terbobot` from ((`2normalisasi` join `bobot`) join `skala`) where `bobot`.`idskala` = `skala`.`idskala` and `bobot`.`idkriteria` = `2normalisasi`.`idkriteria` group by `2normalisasi`.`idnilai`,`2normalisasi`.`idkasus`	SELECT `2normalisasi`.`idnilai` AS `idnilai`,`2normalisasi`.`idkasus` AS `idkasus`,`2normalisasi`.`idkriteria` AS `idkriteria`,`2normalisasi`.`idalternatif` AS `idalternatif`,`2normalisasi`.`idbobot` AS `idbobot`,`2normalisasi`.`idskala` AS `idskala`,`2normalisasi`.`normalisasi` AS `normalisasi`,`skala`.`value` AS `value`,`skala`.`value` * `2normalisasi`.`normalisasi` AS `terbobot` FROM ((`2normalisasi` JOIN `bobot`) JOIN `skala`) WHERE bobot.`idskala` = skala.`idskala` AND bobot.`idkriteria` = 2normalisasi.`idkriteria` GROUP BY `2normalisasi`.`idnilai`,`2normalisasi`.`idkasus`
4	select `3terbobot`.`idkasus` AS `idkasus`,`3terbobot`.`idkriteria` AS `idkriteria`,max(`3terbobot`.`terbobot`) AS `maximum`,min(`3terbobot`.`terbobot`) AS `minimum` from `3terbobot` group by `3terbobot`.`idkriteria`,`3terbobot`.`idkasus`	SELECT 3terbobot.`idkasus` AS `idkasus`, 3terbobot.`idkriteria` AS `idkriteria`,MAX(3terbobot.`terbobot`) AS `maximum`,MIN(3terbobot.`terbobot`) AS `minimum` FROM 3terbobot GROUP BY `3terbobot`.`idkriteria`,`3terbobot`.`idkasus`
5	select `3terbobot`.`idkasus` AS `idkasus`,`3terbobot`.`idalternatif` AS `idalternatif`,sqrt(sum(pow(`4amax_amin`.`maximum` - `3terbobot`.`terbobot`,2))) AS `dplus`,sqrt(sum(pow(`4amax_amin`.`minimum` - `3terbobot`.`terbobot`,2))) AS `dmin` from (`3terbobot` join `4amax_amin`) where `3terbobot`.`idkriteria` = `4amax_amin`.`idkriteria` group by `3terbobot`.`idalternatif`,`3terbobot`.`idkasus`	SELECT 3terbobot.`idkasus` AS `idkasus`, 3terbobot.`idalternatif` AS `idalternatif`,SQRT(SUM(POW(4amax_amin.`maximum` - 3terbobot.`terbobot`,2))) AS `dplus`,SQRT(SUM(POW(4amax_amin.`minimum` - 3terbobot.`terbobot`,2))) AS `dmin` FROM (3terbobot JOIN 4amax_amin) WHERE 3terbobot.`idkriteria` = 4amax_amin.`idkriteria` GROUP BY `3terbobot`.`idalternatif`,`3terbobot`.`idkasus`
6	select `5nilaid`.`idkasus` AS `idkasus`,`5nilaid`.`idalternatif` AS `idalternatif`,`5nilaid`.`dplus` AS `dplus`,`5nilaid`.`dmin` AS `dmin`,`5nilaid`.`dmin` / (`5nilaid`.`dplus` + `5nilaid`.`dmin`) AS `konvensional`,sqrt(pow(`5nilaid`.`dplus` - `5nilaid`.`dmin`,2)) AS `modified` from `5nilaid` group by `5nilaid`.`idkasus`,`5nilaid`.`idalternatif`	SELECT 5nilaid.`idkasus` AS `idkasus`, 5nilaid.`idalternatif` AS `idalternatif`, 5nilaid.`dplus` AS `dplus`, 5nilaid.`dmin` AS `dmin`, 5nilaid.`dmin` / (5nilaid.`dplus` + 5nilaid.`dmin`) AS `konvensional` FROM 5nilaid GROUP BY 5nilaid.`idkasus`, 5nilaid.`idalternatif`

Pertama yang dilakukan adalah membuat rancangan databasenya agar metode TOPSIS bisa diimplementasikan pada query dengan baik tanpa ada redudance data pada setiap langkahnya. Database yang baik merupakan database yang memiliki tingkat redudance data sangat kecil. Berikut gambar 1 merupakan desain database hasil dari rancangan database sebelum metode TOPSIS diimplementasikan pada database MariaDB.

Gambar design database 1 merupakan dasar dari metode TOPSIS yang diimplementasikan menggunakan bahasa query. Hasil implementasi query untuk metode TOPSIS akan berbentuk SQL sep

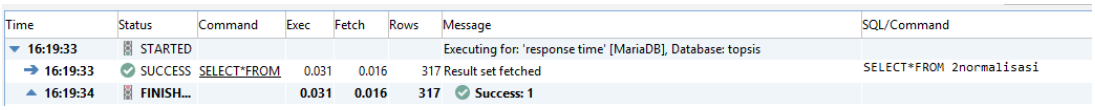
TABEL 8
TABEL JUMLAH DATA UNTUK UJI COBA

No	Uji coba ke-	Jumlah (data)
1	1	144
2	2	228
3	3	342

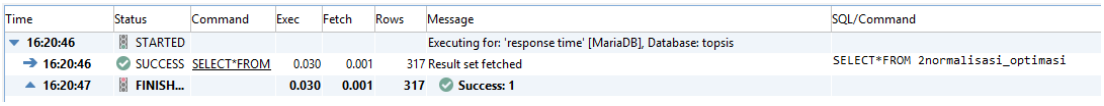
TABEL 9
TABEL UJI COBA KE-1

Langkah TOPSIS	Percobaan						Average	
	1		2		3			
	QNO (second)	QO (second)	QNO (second)	QO (second)	QNO (second)	QO (second)	QNO (second)	QO (second)
1.	0.0046	0.0036	0.0020	0.0019	0.0021	0.0019	0.0029	0.002466667
2.	0.0051	0.0033	0.0036	0.0031	0.0037	0.0030	0.004133333	0.003133333
3.	0.0052	0.0041	0.0060	0.0042	0.0044	0.0038	0.0052	0.004033333
4.	0.0074	0.0064	0.0083	0.0062	0.0065	0.0064	0.0074	0.006333333
5.	0.0144	0.0126	0.0159	0.0128	0.0118	0.0105	0.014033333	0.011966667
6.	0.0108	0.0107	0.0112	0.0109	0.0110	0.0108	0.011	0.0108

QNO = Query Non-Optimization
QO = Query Optimization



Gambar 2. *Response time* Tabel 2normalisasi



Gambar 3. *Response time* Tabel 2normalisasi_optimasi

erti Tabel 7. Pada Tabel 7 berikut ada 5 langkah metode TOPSIS, masing-masing langkah diimplementasikan dengan menggunakan query non-optimization dan query optimization untuk mencari *response time*.

Ujicoba dilakukan menggunakan 3 data kerugian sektor pasca bencana alam dengan jumlah yang berbeda-beda. Karena penelitian ini menguji langkah-langkah metode TOPSIS setelah dilakukan *query optimization*, sehingga diperlukan jumlah data yang berbeda untuk mengetahui *response time* setiap langkah TOPSIS. Setiap data akan di lakukan percobaan sebanyak 3 kali percobaan untuk mendapatkan waktu respon time. *Response time* yang dihasilkan dari 3 kali percobaan tersebut akan di cari rata-ratanya (average) untuk mendapatkan nilai *response time* pada setiap langkah metode TOPSIS pada setiap data ujicoba. Tabel 8 berikut ini merupakan rincian atau detail jumlah data yang di ujicoba.

III. HASIL DAN PEMBAHASAN

Untuk menguji *response time* pada database, penelitian ini menggunakan DbVisualizer 12.0.8 untuk melihat hasil *response time* yang didapat saat mengeksekusi query. Pertama, Dbvisualizer harus terhubung dengan database yang digunakan. Kemudian bekerja dengan menyesuaikan port, nama database, userid dan password database. Setelah berhasil terhubung, maka dapat memanggil tabel yang diinginkan. Misalnya pada Gambar 2 dan Gambar 3 terdapat Tabel 2 normaisasi dan Tabel 2 normalisasi optimasi. Kemudian hasil waktu response ditampilkan di log yang ada waktu *exec* dan *fetch*.

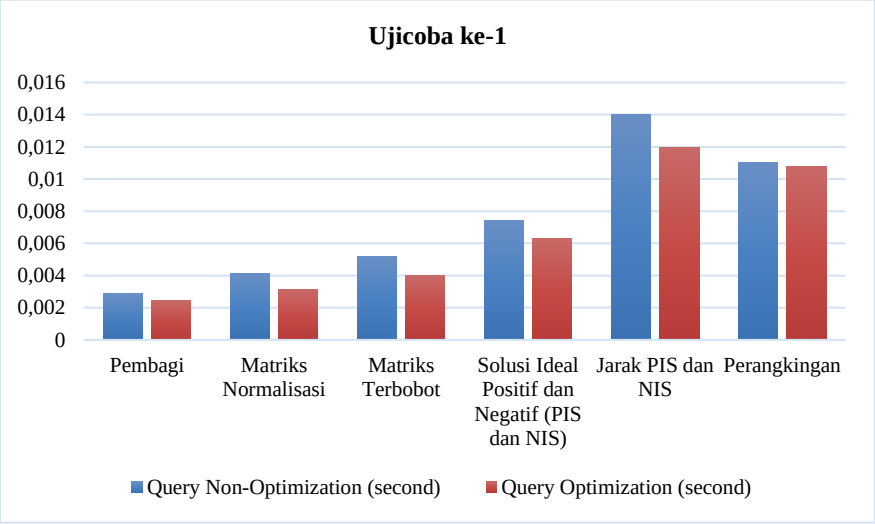
Uji coba ke-1 menggunakan data sebanyak 114 data yang sudah di inputkan pada database MariaDB. Pada Tabel 9 terlihat bahwa ujicoba ke-1 dilakukan 3 kali percobaan yang menghasilkan *respons time* berbeda-beda pada setiap percobaan.

Dari 3 kali percobaan pada ujicoba ke-1 yang berjumlah 114 data diambil rata-rata (average) respons time pada masing-masing langkah metode TOPSIS yang menggunakan QNO (Query Non-Optimization) dan QO (Query Optimization). Hasil dari average masing-masing langkah metode TOPSIS berdasarkan QNO (Query Non-Optimization) dan QO (Query Optimization) dapat dilihat pada Gambar 4.

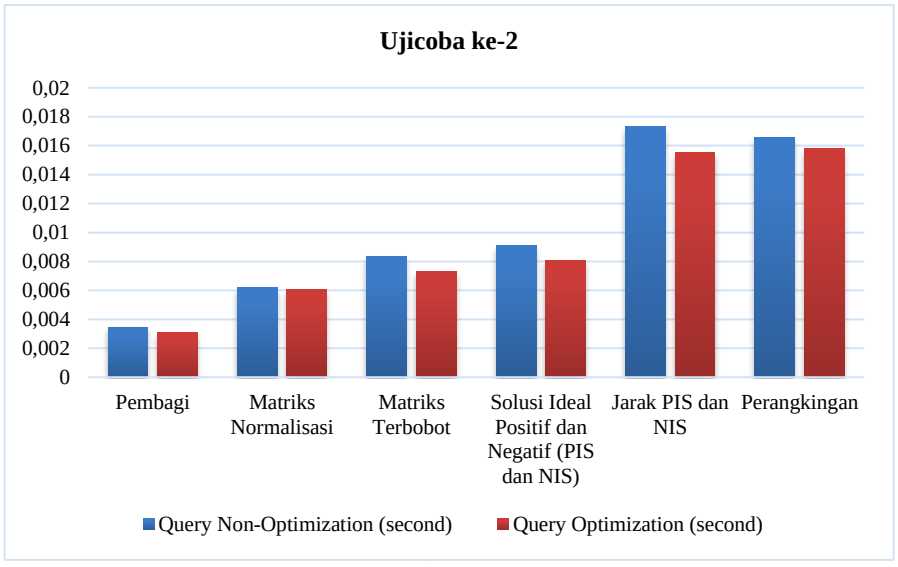
TABEL 10
TABEL HASIL UJI COBA KE-2

Langkah TOPSIS	Percobaan						Average	
	1		2		3			
	QNO (second)	QO (second)	QNO (second)	QO (second)	QNO (second)	QO (second)	QNO (second)	QO (second)
1.	0.0033	0.0030	0.0042	0.0032	0.0029	0.0030	0.003466667	0.003066667
2.	0.0069	0.0058	0.0059	0.0066	0.0058	0.0058	0.0062	0.006066667
3.	0.0076	0.0072	0.0100	0.0074	0.0074	0.0073	0.008333333	0.0073
4.	0.0111	0.0081	0.0079	0.0081	0.0083	0.0081	0.0091	0.0081
5.	0.0157	0.0155	0.0206	0.0159	0.0157	0.0151	0.017333333	0.0155
6.	0.0160	0.0155	0.0177	0.0160	0.0160	0.0159	0.016566667	0.0158

QNO = Query Non-Optimization
QO = Query Optimization



Gambar 4. Grafik Hasil Uji Coba ke-1



Gambar 5. Grafik hasil ujicoba ke-2

Pada Gambar 4 dapat dilihat bahwa pada ujicoba ke-1 dengan menggunakan QO (Query Optimization) dapat mempercepat waktu akses (Respons time) pada proses setiap langkah metode TOPSIS. Mulai dari hasil pembagi dimana antara *response time* QNO dan QO terdapat selisih waktu 0.00043, sampai didapatkan juga *response time* hasil perangkingan dimana *response time* QO lebih cepat 0.0002 dari *response time* QNO.

Ujicoba ke-2 menggunakan data sebanyak 228 data yang sudah di inputkan pada database MariaDB. Pada Tabel 10 dibawah terlihat bahwa ujicoba ke-3 dilakukan 3 kali percobaan yang menghasilkan

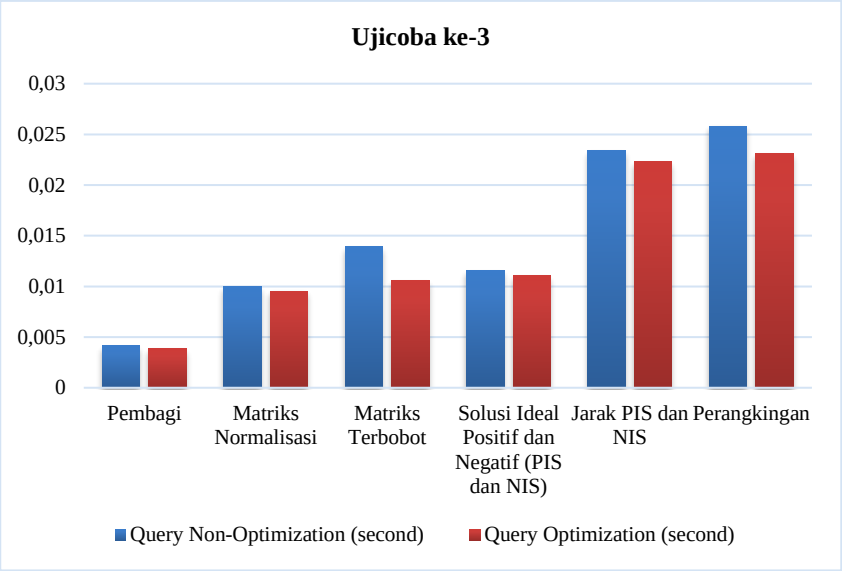
TABEL 11
TABEL UJI COBA KE-3

Langkah TOPSIS	Percobaan						Average	
	1		2		3			
	QNO (second)	QO (second)	QNO (second)	QO (second)	QNO (second)	QO (second)	QNO (second)	QO (second)
1.	0.0040	0.0038	0.0043	0.0038	0.0040	0.0042	0.00416666	0.00386666
2.	0.0093	0.0089	0.0101	0.0099	0.0105	0.0098	0.00996666	0.00953333
3.	0.0203	0.0108	0.0105	0.0105	0.0109	0.0105	0.0139	0.0106
4.	0.0108	0.0108	0.0114	0.0113	0.0125	0.0110	0.01156666	0.01103333
5.	0.0226	0.0220	0.0219	0.0226	0.0257	0.0224	0.0234	0.0223333
6.	0.0312	0.0242	0.0232	0.0225	0.0230	0.0227	0.0258	0.0231333

QNO = Query Non-Optimization
QO = Query Optimization

TABEL 12
HASIL RATA-RATA *RESPONSE TIME*

Langkah Metode TOPSIS	Ujicoba ke-					
	1 (114 data)		2 (228 data)		3 (334 data)	
	QNO (second)	QO (second)	QNO (second)	QO (second)	QNO (second)	QO (second)
Pembagi	0.0029	0.002466667	0.003466667	0.003066667	0.004166667	0.003866667
Matriks Normalisasi	0.004133333	0.003133333	0.0062	0.006066667	0.009966667	0.009533333
Matriks Terbobot	0.0052	0.004033333	0.008333333	0.0073	0.0139	0.0106
Solusi Ideal Positif dan Negatif (PIS dan NIS)	0.0074	0.006333333	0.0091	0.0081	0.011566667	0.011033333
Jarak PIS dan NIS	0.014033333	0.011966667	0.017333333	0.0155	0.0234	0.022333333
Perangkingan	0.011	0.0108	0.016566667	0.0158	0.0258	0.023133333



Gambar 6 Grafik hasil ujicoba ke-3

respons time berbeda-beda pada setiap percobaan.

Dari 3 kali percobaan pada ujicoba ke-2 yang berjumlah 228 data diambil rata-rata (average) respons time pada masing-masing langkah metode TOPSIS yang menggunakan QNO (Query Non-Optimization) dan QO (Query Optimization). Hasil dari average masing-masing langkah metode TOPSIS berdasarkan QNO (Query Non-Optimization) dan QO (Query Optimization) dapat dilihat pada Gambar 10.

Pada Gambar 5 dapat dilihat bawah pada ujicoba ke-2 dengan menggunakan QO (Query Optimization) dapat mempercepat waktu akses (Respons time) pada proses setiap langkah metode

TOPSIS. Mulai dari hasil pembagi dimana antara *response time* QNO dan QO terdapat selisih waktu 0.0004, sampai didapatkan juga *response time* hasil perangkingan dimana *response time* QO lebih cepat 0.000766 dari *response time* QNO.

Ujicoba ke-3 menggunakan data sebanyak 342 data yang sudah di inputkan pada database MariaDB. Pada Tabel 11 dibawah terlihat bahwa ujicoba ke-3 dilakukan 3 kali percobaan yang menghasilkan respons time berbeda-beda pada setiap percobaan.

Dari 3 kali percobaan pada ujicoba ke-3 yang berjumlah 342 data diambil rata-rata (average) respons time pada masing-masing langkah metode TOPSIS yang menggunakan QNO (Query Non-Optimization) dan QO (Query Optimization). Hasil dari average masing-masing langkah metode TOPSIS berdasarkan QNO (Query Non-Optimization) dan QO (Query Optimization) dapat dilihat pada Gambar 4.

Pada Gambar 6 dapat dilihat bawah pada ujicoba ke-3 dengan menggunakan QO (Query Optimization) dapat mempercepat waktu akses (Respons time) pada proses setiap langkah metode TOPSIS. Mulai dari hasil pembagi dimana antara *response time* QNO dan QO terdapat selisih waktu 0.0003, sampai didapatkan juga *response time* hasil perangkingan dimana *response time* QO lebih cepat 0.000266 dari *response time* QNO.

Berdasarkan hasil ujicoba dengan menggunakan 3 data dengan jumlah data yang berbeda-beda dan setiap ujicoba dilakukan 3 percobaan berdasarkan query non-optimization dan query optimization, maka dapat disimpulkan bahwa penerapan query optimization pada metode TOPSIS untuk sistem pendukung keputusan kerusakan sektor pasca bencana sangat bagus karena nilai rata-rata respons time query optimization lebih kecil dibandingkan dengan nilai rata-rata respons time query non-optimization. Pada Tabel 12 berikut merupakan hasil nilai rata-rata respons time dari hasil ujicoba query non-optimization dan query optimization.

Berdasarkan tabel diatas dapat ditarik suatu kesimpulan lagi yaitu semakin banyak data yang digunakan maka nilai *respons time* yang dibutuhkan query non-optimization dan query optimization dalam memproses langkah-langkah metode TOPSIS semakin lama. Tetapi nilai rata-rata respons time query optimization dalam memproses langkah-langkah TOPSIS untuk sistem pendukung keputusan kerusakan sektor bencana sangat bagus karena nilai rata-rata respons timenya lebih kecil jika dibandingkan dengan rata-rata respons time query non-optimization. Oleh karena itu perlu penerapan query optimization pada setiap pembuatan sistem atau aplikasi agar sistem atau aplikasi yang di bangun bisa lebih cepat respons timenya.

Berdasarkan Penelitian sebelumnya yang dilakukan oleh Noviyanti [10], diketahui bahwa query optimization diterapkan pada query database secara umum tanpa metode khusus. Sedangkan pada penelitian ini, query optimization diterapkan pada metode TOPSIS. Hal ini menjadi kelebihan dimana dapat diambil kesimpulan bahwa query optimization tidak hanya bisa diterapkan pada query umum tetapi bisa juga diterapkan pada query metode *Decision Support System* (DSS).

Selain itu, pada penelitian sebelumnya [10], hanya melakukan 1 kali percobaan untuk setiap 3 jumlah data yang berbeda. Sedangkan pada penelitian ini dilakukan uji coba sebanyak 3 kali untuk setiap 3 jumlah data yang berbeda. Hal ini dilakukan untuk mendapatkan hasil yang lebih efektif dengan nilai rata-rata *response time* query optimization lebih kecil daripada query non-optimization.

IV. KESIMPULAN

Berdasarkan hasil penelitian yang didapatkan, dapat disimpulkan bahwa semakin banyak data yang digunakan maka nilai *response time* yang dibutuhkan untuk non optimasi query dan optimasi query dalam memproses langkah-langkah metode TOPSIS semakin lama. Namun nilai rata-rata *response time* query optimasi dalam pengolahan metode TOPSIS memang menggunakan sistem pendukung keputusan kerusakan pasca bencana sangat baik karena nilai *response time* rata-rata lebih kecil jika dibandingkan dengan rata-rata *response time* query non optimasi. Hasil uji coba 3, untuk selisih waktu respon antara query non-optimization dan query optimization pada tahap mencari pembagi adalah 0,0003, maka pada tahap normalisasi matriks adalah 0,000433334 selisih tahap matriks terbobot sebesar 0,0033, kemudian tahap Solusi Ideal Positif dan Negatif sebesar 0,000533334, selisih waktu respon pada tahap PIS dan NIS Distance sebesar 0,001066667, dan perbedaan terakhir pada tahap Ranking yaitu 0,002666667. Di masa mendatang, mengikuti jumlah data yang lebih besar, perbedaan antara waktu respons kueri non-pengoptimalan dan pengoptimalan akan semakin besar. Oleh karena itu perlu diterapkan optimasi query

pada setiap pengembangan sistem atau aplikasi agar sistem atau aplikasi yang kita bangun dapat memiliki *response time* yang lebih cepat. Dengan menerapkan metode *Artificial Intelligence* menjadi lebih dinamis dan memiliki *respons time* yang lebih cepat untuk membangun sistem atau aplikasi. Selain itu, kita dapat menerapkan pengoptimalan query ke database NoSQL.

DAFTAR PUSTAKA

[1] M. S. Yana, L. Setiawan, E. M. Ulfa, and A. Rusyana, "Penerapan Metode K-Means dalam Pengelompokan Wilayah Menurut Intensitas Kejadian Bencana Alam di Indonesia Tahun 2013-2018," J. Data Anal., vol. 1, no. 2, pp. 93–102, 2018, doi: 10.24815/jda.v1i2.12584.

[2] M. Gading Sadewo, A. Perdana Windarto, and A. Wanto, "KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer) Penerapan Algoritma Clustering Dalam Mengelompokkan Banyaknya Desa/Kelurahan Menurut Upaya Antisipasi/ Mitigasi Bencana Alam Menurut Provinsi Dengan K-Means," vol. 2, pp. 311–319, 2018, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/komik.

[3] F. Mahdia, Faya; Noviyanto, "Pemanfaatan Google Maps Api Untuk Pembangunan Sistem Informasi Manajemen Bantuan Logistik Pasca Bencana Alam Berbasis Mobile Web (Studi Kasus : Badan Penanggulangan Bencana Daerah Kota Yogyakarta)," J. Sarj. Tek. Inform., vol. 1, no. 1, pp. 162–171, 2013, doi: 10.12928/jstie.v1i1.2521.

[4] A. Bachriwindi, E. K. Putra, U. M. Munawaroh, and A. T. W. Almais, "Implementation of Web-Based Weighted Product Use Decision Support System to Determine the Post-Disaster Damage and Loss," J. Phys. Conf. Ser., vol. 1413, no. 1, 2019, doi: 10.1088/1742-6596/1413/1/012019.

[5] A. T. Wibowo Almais, M. Sarosa, and M. A. Muslim, "Implementation Of Multi Experts Multi Criteria Decision Making For Rehabilitation And Reconstruction Action After A Disaster," Matics, vol. 8, no. 1, p. 27, 2016, doi: 10.18860/mat.v8i1.3480.

[6] M. Sharma, G. Singh, and R. Singh, "Clinical decision support system query optimizer using hybrid Firefly and controlled Genetic Algorithm," J. King Saud Univ. - Comput. Inf. Sci., 2018, doi: 10.1016/j.jksuci.2018.06.007.

[7] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim, "Approximate query processing using wavelets," VLDB J., vol. 10, no. 2–3, pp. 199–223, 2001, doi: 10.1007/s007780100049.

[8] M. Sharma, G. Singh, and R. Singh, "Design and analysis of stochastic DSS query optimizers in a distributed database system," Egypt. Informatics J., vol. 17, no. 2, pp. 161–173, 2016, doi: 10.1016/j.eij.2015.10.003.

[9] T. Kuo, "A modified TOPSIS with a different ranking index," Eur. J. Oper. Res., vol. 260, no. 1, pp. 152–160, 2017, doi: 10.1016/j.ejor.2016.11.052.

[10] P. Noviyanti, A. Deolika, S. Hartinah, C. A. Haris, T. Maryana, and N. D. Sari, "Perbandingan Query Response time pada Model Query View dan Cross Product," e-Jurnal JUSITI (Jurnal Sist. Inf. dan Teknol. Informasi), vol. 7–2, no. 2, pp. 131–141, 2018, doi: 10.36774/jusiti.v7i2.248.

[11] P. K. Kwok and H. Y. K. Lau, "Hotel selection using a modified TOPSIS-based decision support algorithm," Decis. Support Syst., vol. 120, pp. 95–105, 2019, doi: 10.1016/j.dss.2019.02.004.

[12] I. Warman and R. Ramdaniansyah, "Analisis Perbandingan Kinerja Query Database Management System (DBMS) Antara MySQL 5.7.16 Dan MariaDB 10.1," J. Teknoif, vol. 6, no. 1, pp. 32–41, 2018, doi: 10.21063/jtif.2018.v6.1.32-41.

[13] A. Fauzy, A. Hasani, A. A. Apriyanda, N. Falah, D. Aulia, and A. Sugiarto, "Pembuatan Website : MySQL Database," 2000.

[14] G. L. Ginting, P. N. Dian, and Pristiwanto., "Perancangan Aplikasi Pendeteksi Kesalahan Perintah SQL Query Menggunakan Algoritma Knuth Morris Pratt," J. Ris. Komput. ISSN 2407-389X, vol. 5, no. 4, pp. 377–381, 2018, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom.

[15] M. S. Manahan Siallagan, Mira Kania Sabariah, "Optimasi Query Database Menggunakan Algoritma Genetik," *Issn 1907-5022 Optimasi*, vol. 2008, no. Snati, 2008.