

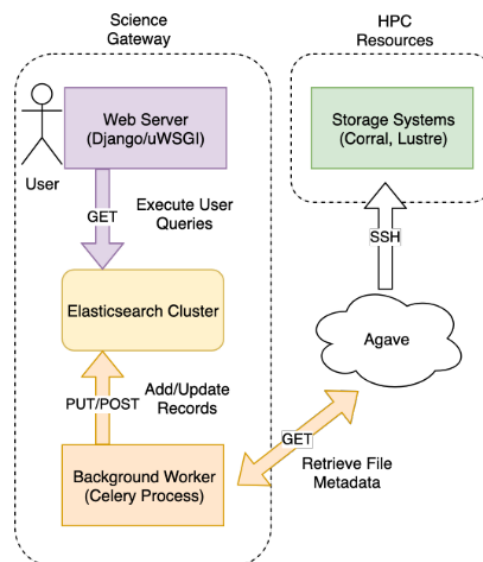
BAB II. TINJAUAN PUSTAKA

Dalam pembuatan penelitian ini penulis telah membaca berbagai jurnal dan penelitian yang pada akhirnya dijadikan referensi dan pembanding dengan penelitian yang dibuat oleh penulis. Pada bagian ini akan terdapat analisis dan penjelasan dari beberapa penelitian-penelitian sebelumnya yang dilakukan oleh peneliti lain, yang mana penelitian-penelitian tersebut memiliki korelasi dengan penelitian ini. Selain itu penelitian-penelitian tersebut akan dijadikan dasar dan patokan bagi penulis dalam menulis penelitian ini.

Penelitian pertama yang menjadi sumber referensi bagi penulis adalah penelitian dengan judul *Leveraging elasticsearch to improve data discoverability in science gateways* yang dibuat oleh J. Rosenber, J. Coronel, J. Meiring, dan kawan-kawan [3]. Penelitian ini berfokus pada analisis penggunaan *Elasticsearch* pada *science gateways* yang mampu menghasilkan solusi pencarian yang dapat meningkatkan *discoverability* dari data. Kemudahan dalam menemukan data merupakan tantangan dalam membangun arsitektur pencarian. Dalam penelitian ini dikatakan bahwa pencarian yang berkualitas baik harus mencakup kemampuan pencarian file dalam skalabilitas ukuran yang besar dan berubah-ubah, pengindeksan yang cepat dan responsif terhadap setiap aktivitas yang dilakukan oleh pengguna, dan perawatan yang mudah dilakukan tanpa perlu pengetahuan khusus mengenai algoritma pencarian.

Pada Gambar 2.1 dijelaskan pengimplementasian *elasticsearch* pada arsitektur pencarian *science gateway*. Saat pengguna melakukan proses pencarian, *query* yang dihasilkan akan diproses melalui *web server*. Setelah itu *query* akan diproses lebih lanjut pada *Elasticsearch cluster*, seluruh proses yang terjadi pada *Elasticsearch cluster* ini dikerjakan secara *background*. Untuk menghasilkan indeks *Elasticsearch* yang *discoverable*, dilakukan pengaturan struktur direktori dari sistem penyimpanan pengguna menggunakan bantuan *Agave API*. *Agave* adalah platform sains yang dikelola oleh The Texas Advanced Computing Center (TACC) yang menyediakan akses *RESTful* ke sumber daya HPC [7]. *Agave* akan

mengambil metadata file dan folder dalam satu direktori sekaligus dengan melakukan koneksi *SSH* ke *HPC* yang berisi data pengguna dan proyek. Metadata ini akan digunakan untuk memperbarui indeks *Elasticsearch*. Selanjutnya akan didapatkan daftar *file* dan *folder* yang ditemukan selama listing awal. Proses rekursif tersebut akan terjadi sampai struktur direktori lengkap dari sistem penyimpanan pengguna telah terindeks.



Gambar 2.1 Arsitektur Pencarian dari science gateway

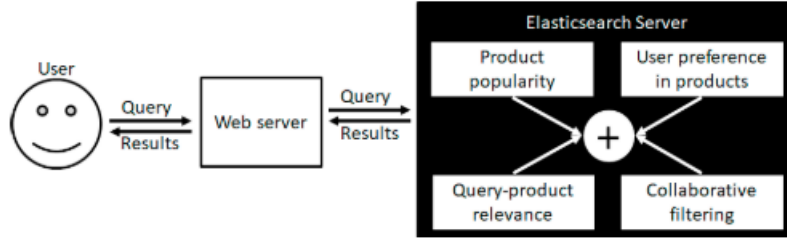
Saat metadata file diindeks ke dalam *Elasticsearch*, setiap *field* akan dianalisis untuk mengisi *inverted index* dengan *token* yang dapat dicocokkan pada saat operasi pencarian dilakukan. Pada pengimplementasian pencarian ini, dilakukan *partial matching* yang difasilitasi oleh *custom-analyzer* yang menggunakan *tokenizer n-gram*. Konfigurasi *tokenizer* dilakukan untuk menghasilkan *token* yang berisi setiap urutan yang berkelanjutan antara dua dan 20 karakter pada nama *file*. Dilakukan penerapan *filter* untuk huruf kecil setiap *token* sehingga pencarian menjadi *case-insensitive*. Perlakuan khusus dilakukan terhadap *query* yang terdiri lebih dari 20 karakter. Pada *query* panjang tersebut akan

diterapkan *filter token* yang memotong kata kunci hingga maksimum 20 karakter, hal ini akan mengakibatkan *query* tersebut cukup hanya perlu mencapai kecocokan pada 20 karakter pertama untuk mengembalikan hit.

Pada penelitian ini [3] didapatkan kesimpulan bahwa *Elasticsearch* mampu menghasilkan solusi dari masalah *discoverability* data yang dialami oleh *science gateway*. Hanya saja mereka belum mengimplementasi fitur *elasticsearch* yang dapat mendukung penyimpanan dokumen secara terdistribusi. Mereka masih menggunakan 1 *node* saja, padahal hal tersebut cukup rawan jika misalnya terjadi kegagalan pada perangkat keras, dan pendistribusian dapat meningkatkan kecepatan pencarian pada saat *volume metadata* yang disimpan meluas ke skala yang lebih besar.

Penelitian selanjutnya yang menjadi referensi bagi penulis adalah penelitian dengan judul *E-commerce Personalization with Elasticsearch* yang ditulis oleh K. Vavliakis, G. Katsikopoulos, A. Symeonidis [2]. Mengaplikasikan personalisasi pada pencarian adalah sebuah tantangan besar, karena harus mempertimbangkan perpaduan yang tepat antara teknologi yang digunakan, data dan konten yang kompleks dan berbeda pada setiap organisasi [2]. Pada penelitian ini penulis mengusulkan sebuah pendekatan baru untuk melakukan pencarian terpersonalisasi menggunakan *Elasticsearch*. Untuk menghasilkan fitur tersebut mereka mengkombinasikan antara pencarian populer, minat pengguna, dan relevansi permintaan produk dengan penyaringan kolaboratif.

Seperti yang sudah disebutkan sebelumnya pada penelitian ini pendekatan dilakukan dengan mempertimbangkan produk, pengguna, dan juga *query* yang telah ditambah dengan informasi dari hasil penyaringan kolaboratif. Arsitektur pencarian yang dibangun pada penelitian ini dapat dilihat pada Gambar 2.2. Setiap data diintegrasikan pada *file json*, yang diimport pada *Elasticsearch*.



Gambar 2.2 Arsitektur Diagram Pencarian Terpersonalisasi yang diusulkan oleh [2]

Perhitungan skor popularitas dilakukan dengan menggunakan rumus di bawah, di mana $buys_i$, $clicks_i$, $views_i$ masing-masing adalah jumlah pembelian, klik, dan dilihat untuk sebuah produk i dan $|buys|$, $|clicks|$, $|views|$ adalah jumlah total pembelian, klik, dan dilihat. Skor popularitas biasanya akan lebih dipengaruhi oleh pembelian, kemudian oleh klik dan terakhir oleh jumlah dilihat, sehingga digunakan w_b , w_c dan w_v .

$$popularity_i = w_b \frac{buys_i}{|buys|} + w_c \frac{clicks_i}{|clicks|} + w_v \frac{views_i}{|views|} \quad (2.1)$$

Selain dari popularitas penelitian ini [2] memperhatikan relasi antara pengguna dan produk. Perhitungan skor relevansi dapat diukur menggunakan rumus di bawah, di mana $buys_{d,u,i}$, $clicks_{d,u,i}$, $views_{d,u,i}$ menggambarkan jumlah pembelian, klik, dan dilihat dari user u untuk sebuah produk i pada hari d , x adalah perbedaan hari antara hari d dan hari pencarian, dan $|buys_u|$, $|clicks_u|$, $|views_u|$ adalah jumlah total pembelian, klik, dilihat oleh user i .

$$\begin{aligned} relevance_{u,i} = & w_b \frac{\sum \left(buys_{d,u,i} * \left(1 + \frac{1}{1 - e^{-x}} \right) \right)}{|buys_u|} \\ & + w_c \frac{\sum \left(clicks_{d,u,i} * \left(1 + \frac{1}{1 - e^{-x}} \right) \right)}{|clicks_u|} \\ & + w_v \frac{\sum \left(views_{d,u,i} * \left(1 + \frac{1}{1 - e^{-x}} \right) \right)}{|views_u|} \end{aligned} \quad (2.2)$$

Salah satu hal lagi yang perlu diperhatikan adalah relevansi *query-product*, yang perhitungannya dapat diukur menggunakan rumus dibawah, di mana

queryNorm adalah ukuran untuk membandingkan *query* ketika terdapat kombinasi jenis *query*, *coord* adalah ukuran kecocokan pada *multiple search*, *tf* adalah ukuran dari jumlah munculnya *t in d*, *idf* adalah ukuran seberapa sering pencarian terjadi di seluruh dokumen, dan *norm* adalah ukuran kecocokan pada *field* yang lebih kecil.

$$\begin{aligned} & \text{queryScore}(q, d) \\ &= \left(\text{queryNorm}(q) * \text{coord}(q, d) * \sum \text{tf}(t \text{ in } d) * \text{idf}(t)^2 \right. \\ & \quad \left. * \text{norm}(t, d) \right) (t \text{ in } q) \quad (2.3) \end{aligned}$$

Pada akhirnya untuk menentukan skor rekomendasi didapatkan rumus berikut yang merupakan kombinasi dari ketiga faktor yang telah dihitung sebelumnya.

$$\begin{aligned} & \text{recommendationScore}_{q,u,i} \\ &= w_p * \text{popularity}_i + w_r * \text{relevance}_{u,i} + w_q \\ & \quad * \text{queryScore}(q, d) + w_s * \text{score}_i \quad (2.4) \end{aligned}$$

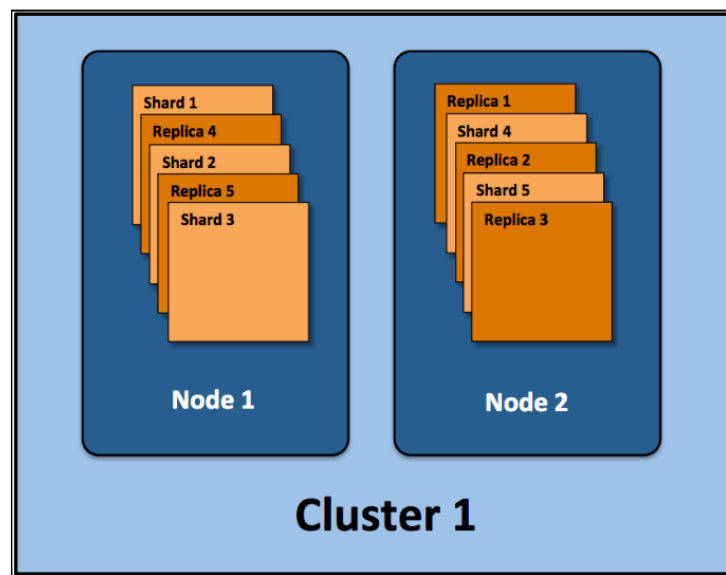
Penelitian ini [2] melakukan uji coba solusi yang mereka usulkan pada *Pharm24.gr* yang merupakan *e-commerce* apotek terbesar di Yunani. Hasilnya menunjukkan bahwa eksperimen mereka dapat meningkatkan *click-through rate* sebesar 4.7%. Selain itu mereka [2] juga melakukan uji *stress test* dengan 5000 klien melakukan *query* terhadap *server*. Hasil tes menunjukkan total waktu respon adalah 224 ms, dengan waktu *query* rata-rata 16ms. Dengan hasil tersebut dapat disimpulkan bahwa penelitian ini tidak hanya berhasil, tetapi bahkan cocok untuk dijadikan pencarian *real-time* di *e-commerce*.

Penelitian selanjutnya yang menjadi referensi bagi penulis adalah penelitian dengan judul *Paper on Searching and Indexing Using Elasticsearch* yang ditulis oleh D. Kalyani dan D. Mehta [4]. Penelitian ini menjadi referensi bagi penulis dalam hal mendalami hal-hal fundamental yang terdapat pada *Elasticsearch* khususnya *searching* dan *indexing*. Menurut [4] pada era seperti sekarang ini sudah tidak dimungkinkan lagi untuk menganalisis data menggunakan *Relational Database Management System (DBMS)*, hal ini terjadi karena pada saat ini perkembangan data terjadi begitu cepat. Oleh sebab itu *Elasticsearch* menjadi

solusi untuk melakukan pencarian data dengan lebih cepat.

Elasticsearch banyak digunakan dalam banyak hal diantaranya adalah *analytics store*, *full text search*, *alerting engine*, *spell checker*, *autocomplete*, dan bahkan untuk menyimpan dokumen seperti biasa [4]. Hal ini memang benar karena berdasarkan pengalaman penulis di Tokopedia, *Elasticsearch* memang banyak dimanfaatkan dalam bidang-bidang yang disebutkan sebelumnya terutama pada *full text search* dan *autocomplete*.

Setiap indeks pada *Elasticsearch* dapat memiliki banyak dokumen, sedangkan di dalam dokumen sendiri dapat terdiri dari lebih dari satu *fields* [4]. Beberapa Node yang bekerja sama disebut sebagai *cluster*, ilustrasi *cluster* sendiri dapat dilihat pada Gambar 2.3. *Elasticsearch* dapat menerima respon cepat dari pencarian karena *Elasticsearch* tidak mencari teks secara langsung, melainkan melakukan pencarian indeks. *Elasticsearch* memiliki kemampuan untuk membagi indeks menjadi pecahan-pecahan yang disebut sebagai *shard*. Ketika sebuah dokumen disimpan dan diindeks, *Elasticsearch* akan mendefinisikan *shard* yang bertanggung jawab terhadap dokumen itu. Secara *default* *Elasticsearch* akan mengindeks setiap *fields* pada dokumen, sehingga *fields* tersebut dapat menjadi *searchable*.



Gambar 2.3 *Elasticsearch Cluster*

Selanjutnya [4] menjelaskan mengenai pembuatan *inverted index* pada *Elasticsearch*. Biasanya jika pengguna misalnya ingin mencari sesuatu tentang apel merah, maka pengguna akan mengetikkan “apel merah” pada pencarian. Jika teknik pengindeksan khusus tidak dilakukan, maka kalimat “apel merah” akan dipindai secara utuh untuk dicari kecocokannya, tentunya hal ini tidak efisien. Hal inilah yang mendukung mengapa *inverted index* diperlukan. Misalnya untuk membuat *inverted index* dari dokumen pada Gambar 2.4. Setiap dokumen akan dibagi-bagi menjadi kata-kata yang terpisah. Setelah itu akan terbentuk daftar yang berisi setiap kata-kata unik, id dokumen di mana kata-kata tersebut terkandung, dan frekuensi kemunculan kata-kata tersebut. Sehingga dari dokumen pada gambar akan terbentuk *inverted index* seperti pada Gambar 2.5.

Document1: “Elasticsearch is fast”

Document2: “Elasticsearch is efficient”

Document3: “Elasticsearch offers high speed”

Gambar 2.4 Contoh dokumen yang diproses pada [4]

Terms	Document	Position	Term Frequency
Elasticsearch	Document1,Document2 , Document 3	1,1,1	3
is	Document1,Document2	2,2	2
fast	Document1	3	1
efficient	Document2	3	1
offers	Document3	3	1
High	Document3	4	1
speed	Document3	5	1

Gambar 2.5 Hasil *inverted index* pada [4]

Dari penelitian yang dilakukan oleh [4] dapat disimpulkan bahwa melakukan pengindeksan pada dokumen dapat menyebabkan proses pencarian menjadi lebih efisien. Dengan penggunaan *inverted index*, *Elasticsearch* dapat dengan handal menangani full text search dengan menghasilkan pencarian yang sangat relevan.

Elasticsearch merupakan *NoSQL database* oleh sebab itu penulis

menjadikan penelitian berjudul *SQL VS NoSQL* yang ditulis oleh D. Mohmmed dan P. Osman [5] menjadi salah satu referensi bagi penulis. Penelitian ini membahas perbandingan antara *SQL* dan *NoSQL* yang tentunya masing-masing memiliki kelebihan dan kekurangannya tersendiri. *NoSQL databases* memiliki banyak fitur canggih di luar fitur dari *RDBMS* konvensional, hal itulah yang menyebabkan “*NoSQL*” secara umum kita kenal sebagai “*Not Only SQL*” [5].

NoSQL adalah jenis basis data yang lahir untuk mengatasi keterbatasan dari basis data relasional yang sudah ada sebelumnya (*RDBMS*). *NoSQL* dapat menangani sejumlah besar data dengan baik yang terstruktur, tidak terstruktur, semi terstruktur dan bahkan hybrid yang mampu ditangani dengan performa yang baik dengan kompleksitas dan biaya yang lebih rendah. Tabel 2.1 berikut merupakan perbandingan antara *NoSQL* dan *SQL* menurut [5].

Tabel 2.1 Tabel perbandingan *NoSQL* dan *SQL* database menurut [5]

<i>NoSQL</i>	<i>SQL</i>
Cara penyimpanan data tidak terstruktur.	Cara penyimpanan data sepenuhnya dilakukan secara terstruktur.
Jumlah data yang disimpan tidak bergantung pada memori fisik dari sistem. Jumlah data bisa diskalakan secara horizontal sesuai kebutuhan	Jumlah data yang disimpan sangat bergantung pada memori fisik dari sistem.
Dapatkan dengan efektif menangani miliaran data.	Kurang efektif menangani data dengan jumlah data yang sangat besar.
Waktu pemrosesan bergantung pada jumlah mesin <i>cluster</i> .	Waktu pemrosesan bergantung pada konfigurasi mesin <i>server</i> .
<i>Schema less</i> .	Penggunaan skema pada database sangatlah ketat.

Fitur-fitur yang dimiliki *NoSQL* berasal dari kebutuhan *e-commerce* dan juga media sosial. Aplikasi-aplikasi ini membutuhkan penyediaan laporan, peringatan, dan integritas data untuk peningkatan pelayanan. *NoSQL* hadir dengan keunggulan yang dapat memenuhi kebutuhan-kebutuhan tersebut diantaranya

adalah skalabilitas, tingkat ketersediaan yang tinggi, kapasitas penyimpanan yang hampir tidak memiliki batas, dan *multitenancy*.

Dari penelitian ini dapat disimpulkan bahwa *NoSQL database* telah hadir dan berhasil memberikan kinerja yang lebih baik dibandingkan dengan *RDBMS* yang telah ada terlebih dahulu. *NoSQL* telah terbukti kemampuannya dalam menangani aplikasi bisnis yang berbasis *real time* dengan analitik yang baik dan juga menjadi sistem pencarian dari suatu perusahaan terkhususnya *e-commerce*.

Penelitian yang dilakukan oleh penulis tentunya memiliki beberapa perbedaan dengan penelitian yang dijadikan referensi oleh penulis. Hal itu terjadi karena setiap penelitian pastinya memiliki tujuannya masing-masing. Tabel 2.2 disajikan untuk melihat perbandingan penelitian yang dilakukan penulis dengan penelitian-penelitian yang ditulis peneliti lain.

Tabel 2.2 Tabel perbandingan penelitian penulis dengan penelitian-penelitian sebelumnya

No	Pembanding	J. Rosenber dan kawan-kawan(2019)	K. Vavliakis dan kawan-kawan(2019)	D. Kalyani dan kawan-kawan(2017)	D. Mohmmmed dan P. Osman (2017)	Penulis (2020)
1	Judul	<i>Leveraging elasticsearch to improve data discoverability in science gateways</i>	<i>E-commerce Personalization with Elasticsearch</i>	<i>Paper on Searching and Indexing Using Elasticsearch</i>	<i>SQL VS NoSQL</i>	Analisis Pemanfaatan NoSQL Database Elasticsearch Pada Mesin Pencarian Tokopedia
2	Tujuan	Menganalisis penggunaan <i>Elasticsearch</i> pada <i>science gateways</i> yang mampu	Menganalisis pendekatan baru untuk melakukan pencarian terpersonalisasi	Menganalisis <i>searching</i> dan <i>indexing</i> pada <i>Elasticsearch</i>	Menganalisis perbandingan antara <i>SQL</i> dan <i>NoSQL</i>	Menganalisis efektivitas penggunaan <i>NoSQL database</i> khususnya

		menghasilkan solusi pencarian yang dapat meningkatkan <i>discoverability</i> dari data	menggunakan <i>Elasticsearch</i>			<i>Elasticsearch</i> pada mesin pencarian Tokopedia
3	Tempat Penerapan	Mesin pencarian <i>science gateways</i>	Mesin Pencarian <i>Pharm24.gr</i>	Tidak disebutkan	Tidak disebutkan	Mesin pencarian Tokopedia