BAB III. LANDASAN TEORI

3.1. Pencarian Data

Proses pencarian adalah sebuah proses yang setiap harinya kita temui dalam kehidupan sehari-hari. Pencarian data yang cepat dan tepat adalah sesuatu yang pastinya kita inginkan dalam setiap kita melakukan proses pencarian. Untuk meningkatkan pengalaman pencarian data tersebut maka munculah SEO (Search Engine Optimization). SEO (Search Engine Optimization) sendiri adalah sebuah mekanisme yang memungkinkan pencari dapat melakukan pencarian data dengan mendapatkan hasil pencarian paling tepat [8]. Oleh sebab itulah pada penelitian ini dilakukan pembuktian *database* mana yang dapat mendukung pencarian yang lebih cepat.

3.2. Availibility

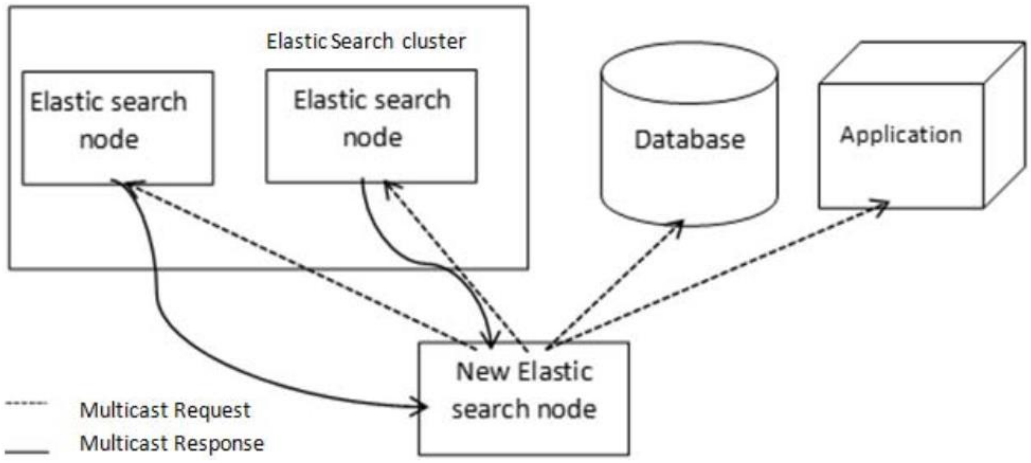
Salah satu aspek *elasticsearch* yang dijadikan pengujian pada penelitian ini adalah *availibility*. *Availibility* adalah kemampuan sejauh mana sistem, subsistem, ataupun segala sesuatu yang berhubungan dengan sistem berada dalam kondisi yang dapat dioperasikan dan berkomitmen dari awal operasi dan ketika kapanpun operasi dipanggil [9]. Kemampuan *availibility* dari *Elasticsearch* pada penelitian ini, akan diuji dari kemampuan suatu *elasticsearch cluster* bekerja apabila beberapa *node* yang ada didalamnya mati.

3.3. Flexibility

Salah satu aspek *elasticsearch* yang dijadikan pengujian pada penelitian ini adalah *flexibility*. *Flexibility* dalam *engineering* mengacu pada kemampuan sistem untuk dapat beradaptasi jika ada perubahan external yang terjadi [10].*Flexibility* yang dimaksud pada penelitian ini adalah kemampuan *elasticsearch* yang dapat dengan fleksibel menambahkan *field* baru tanpa melakukan pengaturan *mapping* terlebih dahulu. Kemampuan *flexibility Elasticsearch* sendiri didukung karena *Elasticsearch* sendiri merupakan *document-oriented database*.

3.4. *Elasticsearch*

Elasticsearch adalah sebuah basis data *NoSQL*, yang dibangun dengan dasar bahasa pemrograman *java* dan menggunakan protokol *HTTP/JSON* [11]. Pencarian pada *Elasticsearch* sangatlah flexibel, serta mendukung dilakukannya *clustering*. Elasticsearch sendiri dibangun dengan basis *Apache lucene*. Di dalam *Elasticsearch* semua yang berhubungan dengan algoritma pencarian dan juga penyimpanan indeks dilakukan oleh *Lucene*. *Elasticsearch* sendiri menyediakan *API* yang lebih fungsional dan *compact*, mendukung skalabilitas, dan memiliki alat operasional yang melapisi implementasi dari penggunaan *Lucene*. Seperti yang dijelaskan pada penelitian milik [11] saat *Elasticsearch node* dibuat, *node* akan menggunakan *discovery module* untuk mencari *node* lainnya yang terletak pada cluster yang sama dan akan saling terhubung satu sama lain seperti yang terlihat pada Gambar 3.1. *Elasticsearch* merupakan teknologi *NoSQL* yang digunakan pada mesin pencarian Tokopedia, oleh sebab itu *Elasticsearch* akan digunakan pada penelitian ini untuk mewakili *NoSQL*.



Gambar 3.1 Proses dimulainya *Elasticsearch* yang dipaparkan pada penelitian [11]

3.4.1. Lucene

Lucene adalah *full text search engine* berbasis *open source* yang dikembangkan oleh *Apache Software Foundatio*n. Menurut penelitian [12]

pencarian pada *Lucene* melibatkan 3 hal utama, yaitu *IndexSearcher*, *TermQuery*, dan *TopDocs*. *IndexSearcher* adalah komponen utama yang dibuat setelah proses pengindeksan. *TermQuery* adalah bagian yang berisi *query*. Sedangkan *TopDocs* adalah bagian yang mengembalikan dokumen yang paling sesuai dengan *query*.

3.4.2. Elasticsearch Index

Menurut penelitian [11] *Elasticsearch Index* adalah tempat dimana data disimpan. *Elasticsearch* sendiri dapat menyimpan data ke lebih dari satu *index*. Sebelumya *index* sering disamakan dengan database dan *type* disamakan dengan *table* seperti yang terdapat pada *relational database*, tetapi pada versi 7.7 *Elasticsearch* telah memberikan penjelasan bahwa *index* dan *database* memiliki konsep yang berbeda [13]. *Elasticsearch* menggunakan *Apache Lucene library* untuk menulis dan membaca data dari *indeks*.

3.4.3 Elasticsearch Node

Setiap kali kita memulai *Elasticsearch Instance*, kita memulai sebuah *node*, penelitian [11] memaparkan bahwa terdapat tiga tipe *Elasticsearch Node*. Pertama adalah *data node* dimana data disimpan dan dicari. Kedua terdapat *master node* yang merupakan *node* yang bertugas untuk mengawasi *cluster* yang mengendalikan *node* lainnya. Ketiga terdapat *tribe node* yang dapat bergabung dengan beberapa *cluster* dan menjadi jembatan di antara *cluster-cluster* tersebut. Pada penelitian ini akan digunakan 3 *node elasticsearch* pada *cluster* yang bernama “tugas-akhir”.

3.4.4 Elasticsearch Cluster

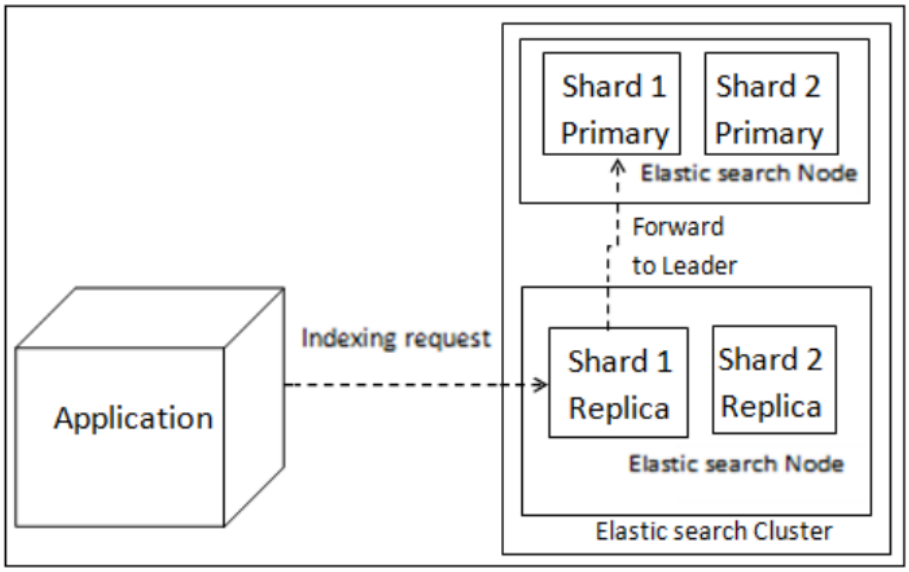
Cluster adalah sekumpulan *Elasticsearch Node* yang bekerja bersama [11]. Sifat *Elasticsearch* yang terdistribusi inilah yang memungkinkan *Elasticsearch* dapat menangani data dalam jumlah yang sangat besar untuk ditangani pada satu beberapa *node* secara langsung. Penelitian ini menggunakan *cluster* dengan tujuan untuk membuktikan *availibility* dari *elasticsearch*.

3.4.5. Elasticsearch Shard

Shard adalah pecahan indeks yang berupa *instance* dari *Apache Lucene* [11]. *Shard* membantu *Clustering* dapat menyimpan data dengan volume yang melebihi kemampuan suatu *server*, untuk itulah *Elasticsearch* menyebarkan data ke beberapa indeks yang disebut sebagai *shard* itu sendiri. Pada penelitian ini akan dibentuk *index* dengan 5 *shard* dan 2 *replication*. Penggunaan *shard* ini juga akan membantu penelitian untuk membuktikan *availibility* dari *Elasticsearch*.

3.4.6. Indexing Data

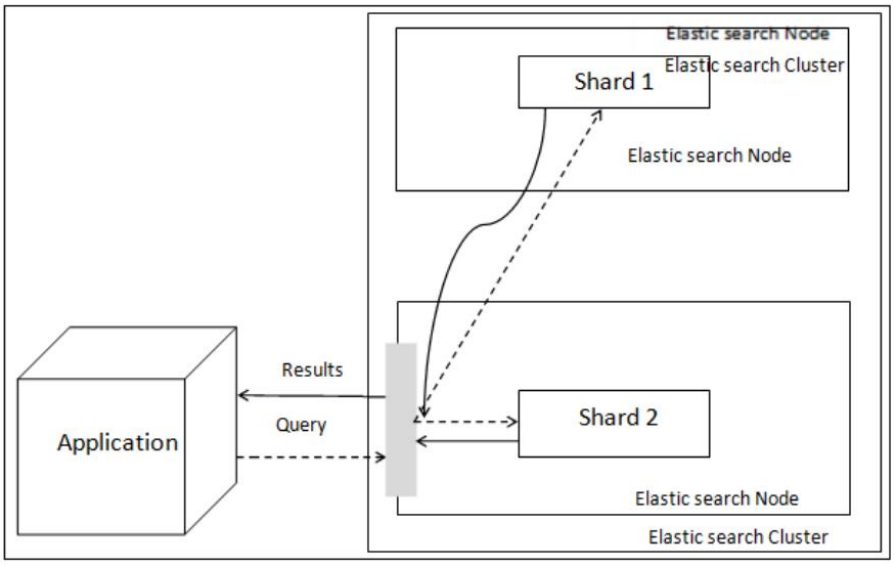
Sebelum sebuah data dapat dicari, perlu dilakukan *indexing* terhadap data. *Indexing* sendiri adalah metode yang dilakukan oleh *Elasticsearch* untuk menginput data. Seperti yang dipaparkan pada penelitian [11] untuk melakukan *indexing* pada *Elasticsearch* terdapat dua cara. Cara pertama dan termudah adalah dengan menggunakan *index API*, yang memungkinkan pengiriman satu dokumen saja ke indeks tertentu. Cara kedua memungkinkan pengiriman banyak dokumen secara langsung menggunakan *bulk API* dan *UDP bulk API*. Pada Gambar 3.2 digambarkan bagaimana proses *indexing* pada *Elasticsearch* dilakukan. *Indexing* adalah teknik yang akan digunakan pada penelitian ini untuk melakukan pengolahan data seperti *insert*, *update*, dan *delete*.



Gambar 3.2 Proses *indexing* pada *Elasticsearch* yang dipaparkan pada penelitian [11]

3.4.7. Querying Data

Query API adalah bagian kunci dari *Elasticsearch API*. Seperti yang tertulis pada penelitian [11] proses *query* dapat dibagi menjadi 2 fase yaitu fase pencar dan fase pengumpulan. Fase pencar berkaitan dengan proses *querying* semua pecahan (*shards*) *index* yang relevan. Fase pengumpulan berkaitan dengan pengumpulan semua pecahan (*shards*) *index* yang relevan, menggabungkannya, menyortir, memproses dan mengembalikan ke *client*. Pada Gambar 3.3 dijelaskan proses *querying data* pada *Elasticsearch*. Penelitian ini akan menggunakan proses *querying data* pada saat eksperimen pencarian data dengan *Elasticsearch*.



Gambar 3.3 Proses *querying* pada *Elasticsearch* yang dipaparkan pada penelitian [11]

3.5. NoSQL Database

Sesuai dengan namanya *NoSQL*, sering direpresentasikan sebagai *No-SQL* atau juga *Not Only SQL* [5]. Sesuai dengan representasi itu maka *NoSQL* adalah sebuah *database* yang didalamnya tidak terdapat perintah-perintah *SQL* dan memiliki konsep penyimpanan yang dapat menangani penyimpanan data yang terstruktur, semi terstruktur, atau bahkan tidak terstruktur [5]. Pada *NoSQL* relasi antar tabel seperti layaknya pada *SQL* tidak diharuskan. Jenis *database* ini sangat cocok untuk pengembangan aplikasi yang memiliki skema yang fleksibel seperti layaknya pada aplikasi modern masa kini. Penelitian ini menggunakan *NoSQL* untuk dibandingkan dengan *SQL*, sebagai bentuk alternatif baru bagi penggunaan *database* untuk tujuan yang lebih spesifik seperti untuk pencarian pada penelitian ini.

3.6. PostgreSQL

Database yang akan digunakan untuk melakukan uji coba dalam penelitian ini, untuk mewakili *SQL database* adalah *PostgreSQL*. *PostgreSQL* sendiri adalah sebuah *open source database* yang dikembangkan oleh sekelompok orang, yang tidak dikembangkan di bawah asuhan sebuah perusahaan atau organisasi tertentu [14]. Walaupun dikembangkan secara *open source*, nyatanya *database* ini mampu

menjadi pelopor beberapa fitur yang kini menjadi fitur standar pada database komersial seperti [14] :

- *Foreign Key*.
- *Query* kompleks.
- *SQL Sub-chooses*.
- *Trigger*.