BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Mesin pencari atau *search engine* adalah program komputer yang digunakan untuk melakukan pencarian situs web. Pada awal kemunculannya, mesin pencari lebih diperuntukkan bagi kalangan akademisi untuk mencari jurnal atau dokumen akademik lainnya. Namun, seiring dengan meluasnya adopsi internet ke berbagai lapisan masyarakat, mesin pencari memiliki peran yang lebih besar dalam penggunaan internet. Mesin pencari berubah menjadi sarana utama dalam pencarian informasi secara umum.

Bila dilihat dari penggunaannya, mesin pencari bekerja dengan cara mengolah *query* yang diberikan oleh pengguna dan menampilkan hasil terkait informasi tertentu di internet yang paling sesuai dengan *query* tersebut. Tetapi mesin pencari tidak dapat langsung mencari halaman secara langsung melalui internet. Mesin pencari membutuhkan database yang berisi informasi pada halaman yang tersebar di seluruh jaringan internet.

Ketika internet masih memiliki lingkup pengguna yang kecil, metode pengumpulan data untuk database mesin pencari masih terbilang primitif. Beberapa orang membuat katalog dari berbagai situs dengan kategori tertentu secara manual dan memperbaruinya dari waktu ke waktu (Seymour et al., 2011). Namun, makin tingginya adopsi internet membuat jumlah data pada internet meledak. Hal ini berakibat pada diperlukannya metode pengumpulan data yang lebih efisien dan metode penerimaan informasi yang lebih cepat.

Mesin pencari membutuhkan serangkaian proses yang perlu dilakukan sebelum dapat menerima *query* terkait informasi tertentu. *Google*, mesin pencari paling populer saat ini, mempunyai beberapa komponen yang menjalankan tugas secara spesifik. Arsitektur *Google* terdiri dari beberapa komponen utama seperti *crawler*, modul *indexer*, modul pemeringkatan *PageRank* dan modul pencari (Brin et al., 1998). Seluruh modul tersebut dibuat secara mandiri tanpa menggunakan aplikasi atau layanan pihak ketiga.

Meski dengan informasi tentang rancangan arsitektur dari publikasi artikel, implementasi ulang arsitektur *Google* secara menyeluruh cukup sulit dilakukan

karena banyak detail dari arsitektur yang tidak dicantumkan. Selain itu *Google* juga banyak membuat implementasi yang telah dioptimalkan untuk kebutuhan mesin pencari baik dari segi performa maupun penggunaan ruang penyimpanannya, tanpa memublikasikan kode program atau algoritma yang digunakan.

Upaya implementasi ulang arsitektur *Google* telah dilakukan (Khatulistiwa 2023) dan menghasilkan integrasi mesin pencari *Telusuri*. Arsitektur tersebut merupakan pengembangan dari penelitian yang telah dilakukan sebelumnya yang menghasilkan *web crawler* (Qoriiba 2021). *Web crawler* bertugas mengumpulkan halaman web berdasarkan *entry point* tertentu. Halaman web tersebut kemudian di ekstrak dan dikumpulkan daftar kata yang termuat pada halaman web tersebut serta *outgoing link* yang merujuk kepada halaman web lain. Kedua data tersebut kemudian disimpan pada database.

Data yang tersimpan pada database selanjutnya di proses oleh modul *PageRank*. Modul *PageRank* menghitung peringkat suatu halaman web berdasarkan seberapa banyak halaman web lain yang merujuk kepada halaman web tersebut. Setelah kalkulasi peringkat halaman, data akan diolah oleh modul *TF-IDF*. Modul *TF-IDF* akan menghitung bobot kata pada dokumen berdasarkan frekuensi kemunculan kata tersebut dalam suatu dokumen tertentu dan jumlah dokumen yang mengandung kata tersebut. Hasil perhitungan skor dari modul *PageRank* dan *TF-IDF* kemudian akan digabungkan dengan menggunakan algoritma *similarity scoring* yang akan menghasilkan skor relevansi suatu halaman.

Arsitektur *Telusuri* saat ini masih memiliki berbagai kekurangan, dan implementasinya juga cukup berbeda dibandingkan dengan arsitektur milik *Google* karena keterbatasan detailnya. Salah satu modul yang belum memiliki implementasi yang sesuai adalah modul *indexing*. *Indexing* adalah suatu proses pemetaan *record* pada database dengan tujuan mempercepat proses pengambilan *record* dari database dan meningkatkan relevansi hasil pencarian. Proses *indexing* akan menghasilkan *index*, yaitu suatu representasi data yang merujuk pada lokasi data yang lebih lengkap. Konsep penggunaan *index* pada mesin pencari sama dengan index yang biasa ditemukan di bagian belakang buku.

Representasi data pada *index* memiliki tingkat akurasi lokasi data (*granularity*) yang dapat diatur, seperti suatu frasa dalam suatu paragraf, atau unit data yang lebih kecil seperti satu kata tertentu saja. Pemilihan tingkat *granularity* dapat memengaruhi performa dan akurasi dari proses pengambilan data. Sebagai contoh, penggunaan *synonym ring* yang dapat melakukan pengelompokan kata yang

bermakna sama seperti *WordNet* dapat memberikan hasil yang lebih baik dibandingkan dengan hanya menggunakan potongan kata biasa (Gonzalo et al., 1998).

Terdapat berbagai implementasi *index* yang disesuaikan dengan jenis data yang disimpan pada database. Mesin pencari membutuhkan implementasi *index* yang dioptimalkan untuk teks secara menyeluruh. Implementasi *index* yang difokuskan kepada penyimpanan teks setidaknya perlu melakukan tiga hal secara efisien. Yang pertama adalah mendukung pengambilan dokumen berdasarkan *query* berupa beberapa kata yang digabungkan oleh operator logika. Yang kedua adalah memiliki kemampuan untuk menambahkan *record* baru secara efisien. Yang ketiga adalah kemampuan untuk membuat pemeringkatan terhadap *record* yang ada apabila tidak ada data yang memenuhi *query* secara penuh (Zobel et al., 1992).

Dari persyaratan di atas, solusi yang umum digunakan untuk database penyimpanan teks adalah *inverted file index* atau biasa disebut *inverted index* saja. Wujud dari *inverted index* adalah sebuah struktur data yang memetakan kosakata dengan dokumen atau teks utama tempat kosakata tersebut berada(Hersh 2001). *Google* juga menggunakan struktur *inverted index* pada implementasi arsitekturnya (Brin et al., 1998).

Proses *indexing* yang akan di replika akan menggunakan detail yang terbatas yang dapat diakses pada *paper Google*. Untuk memroses *query*, *Google* melakukan beberapa langkah berikut. Pertama, *query* dipotong per kata. Setiap kata kemudian di konversi menjadi *wordID*, yang dapat mengidentifikasi suatu kata secara unik.

Dari kumpulan *wordID* tersebut, untuk setiap kata akan dicari kemunculannya pada permulaan daftar *index* yang berisi judul dan *outgoing links*. Pencarian dilakukan dengan proses *scanning* secara menyeluruh pada daftar *index* sampai ditemukan dokumen yang memenuhi seluruh kata pada *query*. Untuk setiap dokumen yang ditemukan, akan dihitung skor dokumen berdasarkan *query* yang ada.

Jika telah sampai pada akhir daftar dokumen dan posisi pencarian berada pada daftar judul dan *outgoing links*, maka posisi akan berpindah ke daftar kata pada seluruh dokumen, dan mulai melakukan pencarian dokumen lagi. Tetapi jika belum mencapai akhir daftar dokumen, maka posisi pencarian akan diulang dari posisi awal dan memulai pencarian lagi. Setelah seluruh daftar *index* dikunjungi, nantinya akan diakhiri dengan mengurutkan dokumen berdasarkan skor.

Penelitian tentang struktur data untuk keperluan *indexing* telah dilakukan dan menghasilkan implementasi modul *indexing* dengan menggunakan struktur

generalized suffix tree (*GST*) (Pratama 2023). Implementasi dari metode *inverted index* dapat menggantikan implementasi modul yang sudah ada secara menyeluruh atau diintegrasikan dengan modul yang sudah ada sebagai bentuk peningkatan kemampuan modul. Modul *indexing* nantinya juga dapat berperan sebagai metode *query ranking* yang berbeda, atau melakukan enkapsulasi nilai *query ranking* yang sudah ada.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang yang di utarakan di atas, maka perumusan masalah pada penelitian ini adalah "**bagaimana meningkatkan performa waktu pencarian dengan menggunakan *index* ?**".

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah pembuatan salah satu komponen dalam arsitektur mesin pencari yaitu modul *indexing*. Sistem *indexing* yang akan dibuat mengacu pada konsep arsitektur *Google* (Brin et al., 1998). Selain itu, implementasi sistem *indexing* ini hanya akan mencakup proses konstruksi awal index berdasarkan data yang sudah dikumpulkan oleh *crawler*.

1.4 Tujuan Penelitian

1. Membuat implementasi *indexing* berdasarkan konsep *inverted index* untuk memenuhi kebutuhan mesin pencari
2. Melakukan integrasi antara struktur *inverted index* dengan *Generalized Suffix Tree* yang sudah dibuat pada penelitian sebelumnya

1.5 Manfaat Penelitian

1. Bagi penulis

   Menambah pengetahuan dibidang *information retrieval* khususnya mengenai *search engine* dan *crawling*, mengasah kemampuan *programming*, dan memperoleh gelar sarjana dibidang Ilmu Komputer. Selain itu, penulisan ini

juga merupakan media bagi penulis untuk mengaplikasikan ilmu yang didapat di kampus ke kehidupan masyarakat.

2. Bagi Universitas Negeri Jakarta

   Menjadi pertimbangan dan evaluasi akademik khususnya Program Studi Ilmu Komputer dalam penyusunan skripsi sehingga dapat meningkatkan kualitas akademik di program studi Ilmu Komputer Universitas Negeri Jakarta serta meningkatkan kualitas lulusannya.