IMPLEMENTASI ALGORITMA BFS (*BREADTH-FIRST SEARCH*) PADA APLIKASI *WEB CRAWLER*

Rizky Tahara Shita[1] , Subandi[2]

[1, 2)]Fakultas Teknologi Informasi, Universitas Budi Luhur
Jl. Raya Ciledug, Petukangan Utara, Kebayoran Lama, Jakarta Selatan 12260
Telp. (021) 5853753, Fax. (021) 5866369
[1]rizky.tahara@gmail.com, [2]subandionline@gmail.com

ABSTRAK

Kebutuhan penyajian informasi sangatlah diutamakan, terutama pada website PT. MP Games yang menyajikan informasi tentang gadget. Saat ini data gadget cukup banyak dan membuat PT. MP Games kesulitan dalam melakukan entry data tersebut, yang saat ini masih dilakukan secara manual; terlebih dengan keterbatasan jumlah karyawan yang melakukan entry data, membuat penyajian informasi menjadi kurang cepat. Pemanfaatan web crawler, merupakan salah satu solusi yang dapat dilakukan adalah dengan membuat proses entry data menjadi otomatis. Dengan adanya web crawler, maka data sumber dapat diambil dari website yang diinginkan untuk bagian data tertentu agar dapat secara otomatis dilakukan pengolahan data untuk penyajian informasi yang lebih baik. Web crawler dengan metode Breadth-First Search ini menjadi dasar aplikasi yang dapat dimanfaatkan oleh aplikasi lainnya, sehingga penyajian informasi menjadi lebih baik. Usulan ini diharapkan dapat membantu PT. MP Games dalam melakukan entry data gadget dengan lebih cepat karena proses otomatisasi yang dilakukan oleh aplikasi, sehingga penyajian informasi pada website PT. MP Games menjadi lebih terkini dan lebih baik dalam penyajiannya.

Kata kunci: *web crawler, web spider, bfs, search engine*

I. PENDAHULUAN

Konten informasi pada saat ini menjadi kebutuhan banyak orang dalam menjalani kegiatan sehari-hari. Seperti bisa kita lihat sendiri, konten informasi berupa berita ataupun artikel-artikel yang menarik dapat menjadi salah satu kebutuhan bagi para konsumennya terhadap konten informasi yang di sajikan. Konten informasi yang disajikan juga ditunjang pada proses penyampaian konten tersebut.

Proses penyampaian konten di internet yang baik merupakan tugas daripada seorang ahli teknik informatika untuk merancang dan menyajikan proses penyampaian konten yang baik melalu situs yang dibuat. Salah satu yang lebih penting ketika kita berbicara konten adalah proses mendapatkan konten itu sendiri.

Proses mendapatkan konten guna memenuhi kebutuhan konten sebuah situs berita atau artikel-artikel dapat ditempuh dengan berbagai cara, salah satunya adalah dengan memanfaatkan aplikasi *web crawler*.

Aplikasi *web crawler* ini merupakan aplikasi yang mungkin tidak banyak digunakan dalam proses mendapatkan konten tetapi mungkin apabila dibangun dengan tujuan yang spesifik akan memiliki keuntungan yang efisien bagi yang menggunakannya sebagai proses mendapatkan konten.

II. LANDASAN TEORI

2.1 Web Crawler

Web Crawler mulai ditulis pada awal tahun 1993, pada tahun ini lahir empat *web crawler* pertama yaitu World Wide Web Wanderer, Jump Station, World Wide Web Worm dan RBSE Spider. Keempat web crawler ini mempunyai fungsi dasar yaitu mengumpulkan informasi dan statistik sebuah *website* dengan menggunakan *seed* URL yang kemudian akan melakukan proses *download* halaman situs tersebut. [1]

Sebuah *web crawler* (juga dikenal sebagai robot atau *spider*) adalah sistem untuk melakukan *download* sebagian besar halaman *web*. *Web crawler* digunakan untuk berbagai tujuan. Tujuan yang paling sering digunakan adalah menjadikan *web crawler* sebagai salah satu komponen utama dari *web search engine* (mesin pencari *web*). [2]

Selain sebagai komponen utama dari *web search engine*, *web crawler* juga bisa digunakan pada aplikasi web pengarsipan, dimana halaman web dengan skala yang besar secara berkala dikumpulkan dan diarsipkan.

2.2 Arsitektur *Web Crawler*

Sebagai awal dari siklus hidup program, program yang mengerjakan proses akan mendapatkan URL dari struktur data *frontier*, yang akan membagi URL sesuai dengan prioritas mereka sesuai dengan kebijakan yang ada. *Thread* pekerja kemudian memanggil *fetcher* HTTP.

Pertama *fetcher* akan melakukan panggilan ke sub-modul DNS untuk menyelesaikan komponen *host* dari URL ke alamat IP dari *web server* yang sesuai (apabila memungkinkan menggunakan hasil *cache* resolusi sebelumnya), dan kemudian menghubungkan ke *web server*, cek untuk setiap kebijakan yang ada, dan kemudian melakukan upaya untuk mengunduh halaman *web*.

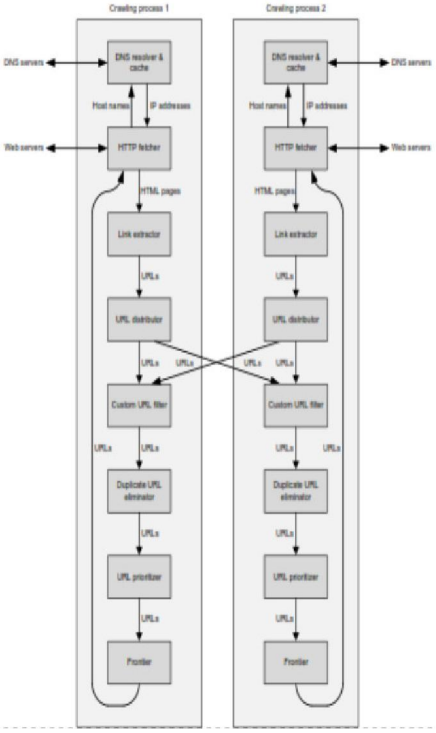
Jika proses unduh berhasil, *web crawler* akan menentukan bahwa halaman *web* memungkinkan atau tidak untuk disimpan dalam repositori halaman *web*. Kemudian halaman akan diteruskan ke pada sub proses *Link Extractor*, yang bertugas melakukan parsing konten HTML halaman dan memisahkan

hyperlink yang terkandung di dalamnya. URL yang sesuai kemudian diteruskan ke distributor URL, yang memberikan tiap URL untuk proses *crawling* berikutnya. [3]

Enam tugas ini biasanya dibuat menggunakan proses *hashing* komponen *host*, *domain*, atau alamat IP-nya (yang terakhir membutuhkan resolusi DNS tambahan) karena pada umumnya *hyperlink* menuju kepada halaman di situs *web* yang sama.

Berikutnya, URL melewati *Custom* URL *Filter* (misalnya, untuk mengecualikan URL situs yang ada pada *black list*, atau URL dengan tertentu ekstensi file yang tidak termasuk dalam tujuan program) dan proses *Duplicate* URL *Eliminator*, yang akan memisahkan URL hanya yang belum pernah dilihat sebelumnya.

Proses yang terakhir adalah URL *Prioritizer* akan memilih posisi untuk URL pada *Frontier*, berdasarkan faktor-faktor seperti pentingnya halaman atau tingkat perubahan.



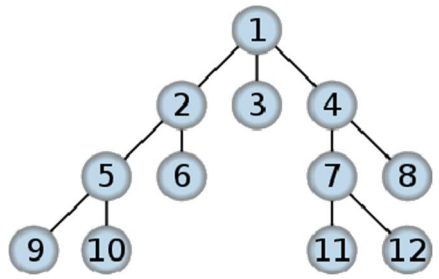
Gambar 1. Arsitektur Web Crawler

2.3 Algoritma BFS (*Breadth-First Search*)

Algoritma *Breadth-First Search* (BFS) atau dikenal juga dengan nama algoritma pencarian melebar adalah sebuah teknik umum yang digunakan untuk melakukan traversal pada graf.

Secara ringkas, algoritma ini memiliki prosedur: traversal dimulai dari simpul, kunjungi semua simpul, kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu, kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul – simpul yang tadi dikunjungi, demikian seterusnya.

Pencarian dalam algoritma BFS dilakukan secara sistematis, artinya biasanya dikunjungi dalam satu arah, misalnya dari simpul paling kiri ke simpul paling kanan. Penelusuran simpul juga dilakukan dalam satu arah terlebih dahulu sebelum mengunjungi simpul pada arah yang lebih tinggi.



Gambar 2. Pohon Pencarian pada Algoritma BFS

Dalam implementasinya, algoritma BFS memerlukan matriks ketetanggaan A = [aij] yang berukuran n × n , antrean q untuk menyimpan simpul yang telah dikunjungi, dan tabel boolean dikunjungi. Pseudo-code dari algoritma BFS adalah sebagai berikut:

```
prosedur BFS(input v:integer)
  Deklarasi
    w : integer
    q : antrean
    prosedur BuatAntrean(input/output q : antrean)
    prosedur MasukAntrean(input/output q:antrean, input
v:integer)
    prosedur HapusAntrean(input/output q:antrean, output
v:integer)
    function AntreanKosong(input q:antrean) -> boolean

  Algoritma
    BuatAntrean(q)
    write(v)
    dikunjungi[v] <- true

    MasukAntrean(q, v)

    while not AntreanKosong(q) do
      HapusAntrean(q, v)
      for tiap simpul w yang bertetangga dengan simpul v
do
        if not dikunjungi[w] then
          write(w)
          MasukAntrean(q,w)
          dikunjungi[w] <- true
        endif
      endfor
    endwhile
```

III. ANALISA MASALAH DAN PERANCANGAN

3.1 Analisa Masalah

Berdasarkan kebutuhan pengguna (*user requirements*) pada PT MP Games, dapat disimpulkan bahwa masalah yang dihadapi oleh instansi tempat penulis melakukan riset adalah kurangnya kebutuhan mengisi konten yang memuat spesifikasi dari setiap *gadget* baik yang beredar di pasar saat ini maupun gadget yang sudah tidak beredar di pasar saat ini pada situs

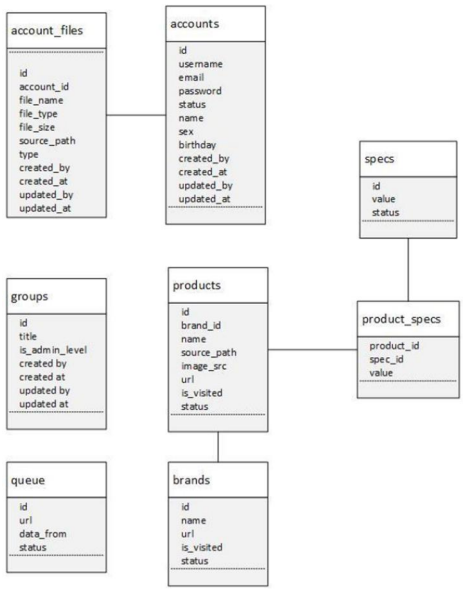
https://jalantikus.com yang dikelola oleh PT MP Games. Konten yang berisi spesifikasi pada bagian *gadget* di situs https://jalantikus. com inipun masih dimasukkan secara manual oleh bagian pengisi konten. Proses input konten yang manual dapat memakan lebih banyak waktu, sehingga membuat pekerjaan menjadi lebih berat dimana seorang pengisi konten harus mencari konten serupa dahulu kemudian baru diinputkan.

Salah satu solusi yang mungkin diterapkan guna memecahkan masalah tersebut adalah dengan membangun aplikasi *web crawler* menggunakan metode *Breadth First Search* (BFS), dimana sebuah aplikasi *web crawler* ini akan bekerja untuk mendapatkan konten spesifikasi *gadget* dari situs http://www.gsmarena.com secara keseluruhan yang kemudian akan digunakan untuk mengisi konten dibagian *gadget* pada situs https://jalantikus.com. Pengisian konten dengan konsep ini dapat mengurangi beban pekerjaan sang pengisi konten spesifikasi *gadget* pada situs https://jalantikus. com. Penggunaan waktu dalam pengisian konten juga dapat lebih efisien dikarenakan jumlah konten yang masuk lebih banyak, namun waktu yang digunakan lebih singkat dari cara manual.

3.2 Perancangan

A. Struktur Basis Data

Struktur basis data yang digunakan dapat dilihat pada *Class Diagram* sebagai berikut:



Gambar 3. Class Diagram

B. Rancangan Basis Data

Adapun spesifikasi basis data yang digunakan adalah sebagai berikut:

Table 1. Rancangan Basis Data Tabel Account Files

Nama Field	Tipe Data	Ukuran	Keterangan
id	int	11	Id
account_id	int	11	Id account
file_name	varchar	255	Nama file
file_type	varchar	50	Tipe file
file_size	float	10	Ukuran file
source_path	varchar	255	Path file
type	varchar	35	Tipe
created_by	varchar	20	Dibuat oleh
created_at	timestamp	10	Dibuat pada
updated_by	varchar	20	Diperbarui oleh
updated_at	datetime	19	Diperbarui pada

Table 2. Rancangan Basis Data Tabel Account

Nama Field	Tipe Data	Ukuran	Keterangan
id	int	11	Id
username	varchar	255	Username
email	varchar	255	Email
password	varchar	40	Password
status	varchar	1	Status
name	varchar	255	Nama
sex	varchar	1	Jenis Kelamin
birthday	date	10	Tanggal Lahir
created_by	varchar	20	Dibuat oleh
created_at	timestamp	10	Dibuat pada
updated_by	varchar	20	Diperbarui oleh
updated_at	datetime	19	Diperbarui pada

Tabel 2. Rancangan Basis Data Tabel Brands

Nama Field	Tipe Data	Ukuran	Keterangan
id	int	11	Id
name	varchar	70	Nama merk
url	varchar	255	Link
is_visited	int	1	Sudah dikunjungi
status	char	1	Status

Tabel 3. Rancangan Basis Data Tabel Groups

Nama Field	Tipe Data	Ukuran	Keterangan
id	int	11	Id
title	int	11	Judul
is_admin_level	varchar	70	Level admin
created_by	varchar	20	Dibuat oleh
created_at	timestamp	10	Dibuat pada
updated_by	varchar	20	Diperbarui oleh
updated_at	datetime	19	Diperbarui pada

Tabel 4. Rancangan Basis Data Tabel Products

Nama Field	Tipe Data	Ukuran	Keterangan
id	int	11	Id Produk
brand_id	int	11	Id Merk
Name	varchar	70	Nama Produk
source_path	varchar	255	Lokasi Foto

image_src	varchar	255	Link Foto
url	varchar	255	Link Produk
is_visited	int	1	Sudah dikunjungi
status	char	1	Status

Tabel 5. Rancangan Basis Data Tabel Specs

Nama Field	Tipe Data	Ukuran	Keterangan
id	int	11	Id
value	varchar	70	Value Spesifikasi
status	char	1	Status

Tabel 6. Rancangan Basis Data Tabel Products Specs

Nama Field	Tipe Data	Ukuran	Keterangan
product_id	int	11	Id produk
spec_id	int	11	Id spesifikasi
value	varchar	250	Spesifikasi Produk

Tabel 7. Rancangan Basis Data Tabel Queue

Nama Field	Tipe Data	Ukuran	Keterangan
id	int	11	Id
url	varchar	250	Link yang akan dikunjungi
data_from	varchar	35	Keberadaan data
status	int	1	Status

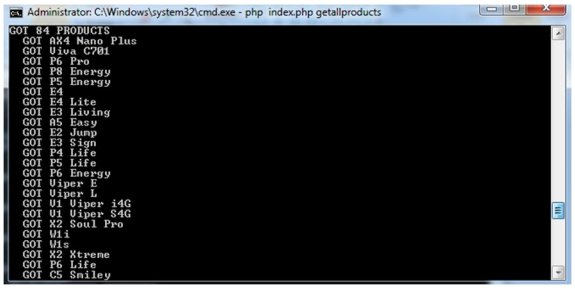
IV. IMPLEMENTASI

Agar dapat melihat bagaimana berjalannya aplikasi *web crawler* yang telah dibuat serta bagaimana contoh pemanfaatannya, maka pada bagian ini disertakan beberapa tampilan layar pada program yang telah dibuat. Tampilan layar dari program yang telah dibuat sendiri dibagi menjadi dua bagian yaitu:

4.1 Crawler

A. Proses getAllProducts

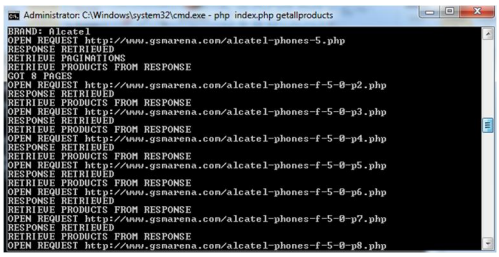
Kondisi aplikasi saat ini menunjukkan bahwa aplikasi ini telah melakukan proses getAllProducts yang sedang berjalan dan telah memasukkan data yang telah diperoleh ke dalam *database*.



Gambar 4. Proses getAllProducts

B. *Proses Pagination*

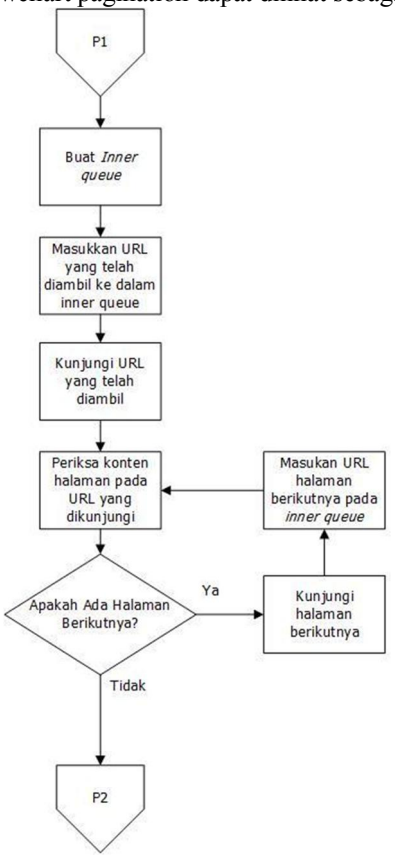
Kondisi aplikasi ini menunjukkan bahwa aplikasi ini telah melakukan sub proses dari proses getAllProducts yang sedang berjalan dan telah berhasil membaca seluruh halaman yang kemudian akan dimasukkan ke dalam *database*.



Gambar 5. Proses Pagination

C. *Flowchart Pagination*

Proses Pagination disini merupakan subproses daripada proses getAllProducts, proses Pagination ini bertujuan untuk membaca halaman pada URL merk yang telah didapat guna mendapatkan seluruh list produk *gadget*. Proses ini diperlukan; karena pada URL merk, ada yang memiliki lebih dari satu halaman. Flowchart pagination dapat dilihat sebagai berikut:

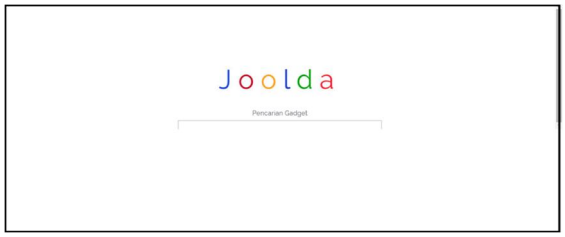


Gambar 6. Flowchart Pagination

4.2 Pemanfaatan *Crawler*

4.2.1 Sisi User

Pada tampilan layar *home search engine*, bisa dilihat bahwa aplikasi *search engine* sudah siap melakukan pencarian berdasarkan *keyword* yang dimasukkan pada *textbox* yang tersedia.

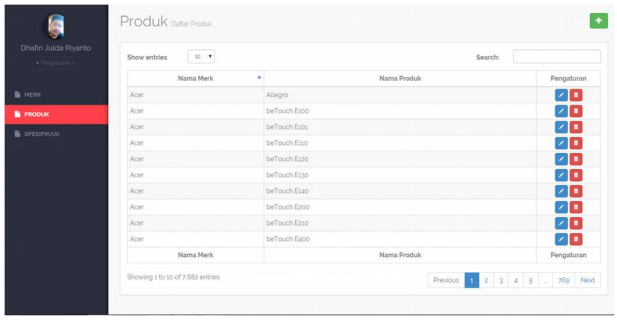


Gambar 7. Pemanfaatan Hasil Crawler pada Search Engine

4.2.2 Sisi Admin

A. *Master* Products

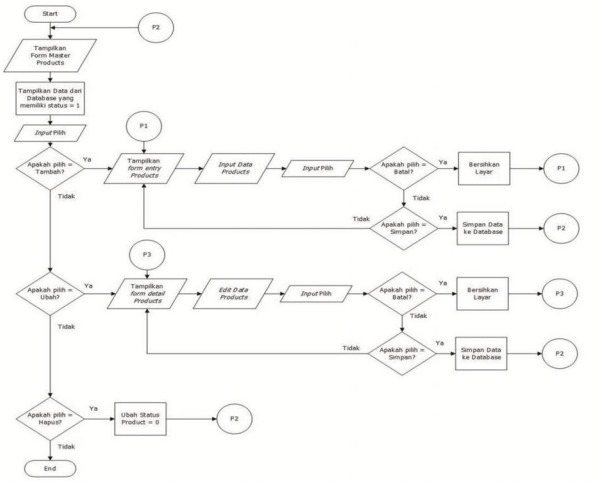
Tampilan layar Form Master Products merupakan tampilan layar yang bertujuan untuk menambah, mengubah dan menghapus data produk *gadget* yang telah didapat dari aplikasi *web crawler*.



Gambar 8. Master Products

B. Flowchart *Master* Products

Flowchart master *products* dapat dilihat pada gambar berikut ini:



Gambar 9. Flowchart Master Products

V. PENUTUP

A. Kesimpulan

Kesimpulan yang dapat diambil dari aplikasi *web crawler* dengan menggunakan algoritma BFS (*Breadth-First Search*) serta contoh aplikasi pemanfaatannya dalam bentuk aplikasi *Search Engine* adalah:

- Algoritma BFS merupakan metode yang efektif digunakan dalam pembentukan aplikasi *web crawler*. Dilihat dari cara kerja a plikasi *web crawler* dan cara kerja algoritma BFS itu sendiri dapat dikatakan kebutuhan pencarian data pada halaman web.
- Aplikasi *web crawler* yang dijalankan pada *local web server* tentunya akan tidak berjalan maksimmal pada kecepatan akses penjelajahan di internet jika dibandingkan dengan aplikasi *web crawler* yang dijalankan melalui *online* pada *server live*.
- Pembangunan aplikasi *web crawler* menggunakan bahasa pemrograman PHP dinilai kurang efektif karena bahasa pemrograman PHP memiliki waktu *timeout* tertentu yang akan membatasi proses *crawling*.

B. Saran

Adapun saran yang mungkin diberikan pada pembangunan aplikasi *web crawler* dengan implementasi algoritma BFS antara lain:

- Untuk bahasa pemrograman, aplikasi web crawler tidak seharusnya dibangun menggunakan bahasa pemrograman PHP karena masih terkendala masalah timeout, memang ada cara yang dilakukan untuk mengatasi masalah ini namun aplikasinya akan berjalan sedikit kurang kondusif. Jadi mungkin lebih baik aplikasi web crawler dibangun menggunakan bahasa pemrograman yang lebih ringan.

- Untuk metode yang digunakan saran penulis adalah tetap menggunakan algoritma BFS, karena algoritma BFS dinilai paling efisien digunakan sebagai metode dasar pembuatan *web crawler*.

DAFTAR PUSTAKA

[1] Mirtaheri, Seyed M, et. al., *A Brief History of Web Crawlers*, CASCON, 2013.

[2] Olston, Christoper, Najork, Mark, *Web Crawling*. Foundations and Trends® in Information Retrieval. Vol. 4, No. 3. Canada, 2010

[3] Choudhary, Suryakant, et. al., *Crawling Rich Internet Applications: The State of the Art*, CASCON, 2012