Jurnal Rekursif, Vol. 9 No. 1 Maret 2021, ISSN 2303-0755
http://ejournal.unib.ac.id/index.php/rekursif/

STUDI PERBANDINGAN IMPLEMENTASI *STRING MATCHING* DENGAN METODE *SEQUENTIAL SEARCHING* DAN KONDISI *LIKE* PADA PENCARIAN JUDUL SKRIPSI

Ferzha Putra Utama[1], Raden Muhammad Hilmi Nurhadi[2], Devina Fitria[3], M. Panji Ramadhan[4]

[1,2,3]Program Studi Infomatika, Fakultas Teknik, Universitas Bengkulu.
Jl. WR. Supratman Kandang Limun Bengkulu 38371A INDONESIA
(telp: 0736-341022; fax: 0736-341022)

[1]fputama@unib.ac.id
[2]hilminurhadi2@gmail.com
[3]devinafitria1@gmail.com
[4]rpanji28.19@gmail.com

Abstrak: Metode pencocokan string banyak digunakan pada fungsi pencarian dalam sebuah sistem informasi. Salah satu sistem informasi yang membutuhkan metode ini adalah sistem informasi pengajuan judul skripsi, dikarenakan judul skripsi yang diajukan harus bersifat unik atau tidak boleh sama dengan judul skripsi yang sudah ada. Penelitian ini melakukan studi perbandingan terhadap dua buah metode *exact string matching* yaitu dengan *sequential searching* dan kondisi Like pada SQL yang diterapkan pada fitur pencarian judul skripsi pada sistem informasi pengajuan judul skripsi Program Studi Sistem Informasi Fakultas Teknik, Universitas Bengkulu. Kedua metode ini mencocokkan *string* pada record yang telah ditentukan berdasarkan *keyword* yang di-input. *Sequential searching* mencocokkan data dengan membuat *array* pada semua data judul skripsi yang telah diajukan, sedangkan kondisi Like membaca semua data pada tabel judul skripsi. Hasil penelitian menununjukkan, kedua metode menunjukkan hasil yang sama pada dalam pencocokan data yang relatif kecil. Kedua metode tersebut kurang efektif digunakan pada data yang besar karena membaca semua data tanpa penanda lain kecuali *keyword*.

Kata kunci: skripsi, *string matching*, *sequential searching*, kondisi like.

Abstract: ***String matching method is widely used for search function of any information systems. One of them is information system for the submission of the thesis topic, due to the topic must be unique or it can't be the same as the existing topic. This research conduct a comparative study of two exact string matching methods, sequential searching and SQL Like condition which were applied to the thesis topic search feature in the Information System Study Program of Engineering Faculty, University of Bengkulu. These method match string of the specified record based on input keyword. Sequential searching matches the data by creating an array of all the submitted thesis topic, while the Like condition reads all the data in the table. The results show both methods have the same results in matching relatively small data. Both of these method are less effective in using large data due to they read all data without other markers except the keyword.***

Keywords: thesis, string matching, sequential seacrhing, like condition

I. PENDAHULUAN

Skripsi merupakan tugas akhir mahasiswa Sarjana Strata-1 berbentuk karya tulis ilmiah yang berisi tentang penjelasan hasil penelitian. Mahasiswa yang ingin mengajukan judul skripsi dan dosen yang memeriksa proposal skripsi mahasiswa masih dilakukan secara manual. Maka dari itu proses pengajuan judul skripsi memakan waktu yang lama.

Dalam penelitian ini sistem informasi yang dikembangkan dibangun dengan bahasa pemrograman php dengan *Data Base Managemen System* (DBMS) *open source* MySQL. Sistem ini mampu menangani pengajuan judul skripsi mahasiswa pada Program Studi Sistem Informasi, Fakultas Teknik Universitas Bengkulu. Pentingnya sistem informasi ini adalah untuk mendata mahasiswa dengan judul skripsi yang diajukan, merekam *history* pengajuan judul hingga mampu mendeteksi kesamaan judul yang telah/pernah diajukan sebelumnya.

Lebih lanjut penelitian ini akan membandingkan dua metode yang diterapkan dalam fitur pencarian judul skripsi pada sistem informasi pengajuan skripsi Program Studi Sistem Informasi Universitas Bengkulu. Metode yang dibandingkan adalah pencocokan *string* menggunakan sintaks SQL "Like" dengan *sequential searching.* Tujuannya adalah untuk mengetahui keakuratan pencocokan *string* pada judul skripsi yang diajukan mahasiswa.

II. METODE PENELITIAN

A. Sistem Informasi

Sistem Informasi adalah kumpulan atau susunan yang terdiri dari perangkat keras dan perangkat lunak serta tenaga pelaksanaannya yang bekerja dalam sebuah proses berurutan dan secara bersama-sama saling mendukung untuk menghasilkan suatu produk (Dengen, 2009:48).

B. String Matching

Proses pencocokan string termasuk ke dalam banyak area dalam bidang komputer. String matching dapat diterapkan untuk pengambilan informasi, analisis informasi, dan berbagai variasi lainnya dalam implementasi perangkat lunak [1]. *String matching* merupakan pencarian semua kemunculan *query* yang selanjutnya disebut *pattern* ke dalam *string* yang lebih panjang [2]. *String matching* dirumuskan dengan:

$$x = x\,[0...m-1] \text{---- (1)}$$
$$y = y\,[0\ldots n-1] \text{---- (2)}$$

dengan:
x = *pattern*
m = panjang *pattern*
y = *text*
n = panjang *text*

Masalah yang sering terjadi dalam pencocokan *string* (*string matching*) adalah mencari *string* yang terdiri dari beberapa karakter dalam sejumlah *text*. Proses pencarian *string* berperan penting dalam mendapatkan dokumen yang sesuai dengan kebutuhan informasi. Pencocokan *string* secara umum dibedakan menjadi dua yaitu pencocokan string secara *exact string matching* (sama persis) dan *inexact string matching* (berdasarkan kemiripan) [3].

C. *Sequential Searching*

Sequential seacrhing method merupakan teknik pencarian beruntun yang dalam penelitian ini digunakan untuk pencarian data dalam *array*. *Array* tersebut berisi semua *record* (data) pada tabel judul skripsi, yang kemudian akan dicocokkan dengan *keyword* yang di-*input* oleh *user* dari awal hingga akhir, begitu seterusnya hingga data yang dicari telah ditemukan [4]. Jika perbandingan bernilai sama, maka pencarian dihentikan dan dinyatakan sukses. Sedangkan apabila perbandingan tidak bernilai sama maka,

a. Jika data tidak terurut (data acak), maka pencarian akan dilanjutkan ke data selanjutnya.
b. Jika data terurut secara menaik (*ascending*), maka pencarian hanya akan dilanjutkan ke data selanjutnya yang berada di sebelah kanan data yang sedang dibandingkan apabila data yang dicari (X) lebih besar dari pada yang sedang dibandingkan sekarang.
c. Jika data terurut secara menurun (*descending*), maka pencarian hanya akan dilanjutkan ke data selanjutnya yang berada di sebelah kanan data yang sedang dibandingkan apabila data yang dicari (X) lebih kecil daripada data yang sedang dibandingkan sekarang.

Secara umum formula yang digunakan dalam sequential searching adalah sebagai berikut: terdapat L yang merupakan larik yang berisi *n* data (L[0], L[1],…,L[*n*-1] dan *k* adalah data yang henda dicari, pencarian dilakukan untuk menemukan:

$$L[i] = k \text{ --- } (3)$$

Dengan *i* adalah nilai indeks terkecil yang memenuhi kondisi $0 \leq k \leq n\text{-}1$ [5]. Algoritme pencarian *linier* adalah bagian perulangan (*loop*) yaitu *While* dengan dua kondisi yakni ≤ yang mengontrol agar perulangan jangan sampai melewati batas dan *Not* (ditemukan) yang mengontrol pencarian apabila data sudah ditemukan maka pencarian tidak perlu lagi dilanjutkan. Jadi, hal yang mengakibatkan proses pencarian keluar dari bagian perulangan adalah barisan sudah habis yakni ≥ atau data yang dicari sudah ditemukan yakni ketemu = *True*.

Dalam sistem informasi pengajuan judul skripsi perintah yang digunakan untuk pencarian judul dengan metode sequential searching ditunjukkan pada Gambar 1.

```php
<?php
$username = session()->get('username');
$db = \Config\Database::connect(); //Koneksi database di codeigniter 4
$akun = $db->query("SELECT * FROM tb_pengajuan"); //Query memilih tabel database
$nrow = $akun->getResultArray(); //Mengambil semua data dalam bentuk array dari tb_pengajuan

$n = count($nrow); //Menghitung banyaknya data

for ($i = 0; $i < $n; $i++) { //Melakukan looping sebanyak data yang diambil tadi
    $datajudul = $nrow[$i]['judul']; //Mengambil judul dari semua data yang ada
    $posisi = stripos($datajudul, $judul); //Melakukan pengecekan judul

    if ($posisi !== FALSE) { //Jika terdapat judul yang sama, maka kode dibawah ini akan dijalankan
?>
        <tr>
            <td class="align-middle text-center"><?= $i; ?></td>
            <td class="align-middle"><?= $nrow[$i]['npm']; ?></td>
            <?php
            $npm = $nrow[$i]['npm'];
            $db = \Config\Database::connect();
            $akun = $db->query("SELECT * FROM tb_mahasiswa WHERE npm = '$npm'");
            $row = $akun->getResultArray();
            $nama = isset($row[0]['nama_mhs']) ? $row[0]['nama_mhs'] : '';
            ?>
            <td class="align-middle"><?= $nama; ?></td>
            <td class="align-middle"><?= $datajudul; ?></td>
            <td class="align-middle"><a href="/file_proposal/<?= $nrow[$i]['proposal']; ?>">Download</a></td>
            <?php if ($nrow[$i]['status'] == '') { ?>
                <td class="align-middle text-center">
                    <font color="gray"><i class='fa fa-spinner'></i></font>
                </td>
            <?php } else if ($nrow[$i]['status'] == 'n') { ?>
                <td class="align-middle text-center">
                    <font color="red"><i class='fa fa-times'></i></font>
                </td>
            <?php } else if ($nrow[$i]['status'] == 'y') { ?>
                <td class="align-middle text-center">
                    <font color="green"><i class='fa fa-check'></i></font>
                </td>
            <?php } ?>
            <td class="align-middle"><?= $nrow[$i]['created_at']; ?></td>
        </tr>
<?php
    } else {
    }
}
?>
```

Gambar 1. Code sequential searching

D. *Like Condition*

Kondisi Like merupakan suatu metode dalam pencocokan pola. Berbeda dengan operator persamaan (=) pada sintaks SQL umumnya yang mengharuskan kondisi sama persis dengan satu nilai karakter dengan yang lain, Like mencocokkan sebagian dari satu nilai karakter ke nilai lainnya dengan mencari nilai pertama untuk pola yang ditentukan oleh nilai kedua. Like menghitung string menggunakan karakter seperti yang ditentukan oleh set karakter input [6].

Kondisi Like merupakan suatu fungsi pengganti karakter dalam sintaks SQL. Kata pengganti dengan kondisi Like pada SQL terdapat dua macam, yakni dengan menggunakan "%" dan "_". Keduanya memilki fungsi yang berbeda. Dalam penelitian ini akan digunakan kondisi Like dengan %. Perintah Like yang diterapkan ditunjukkan pada Gambar 2.

```
public function cekPengajuan($judul = false)
{
    if ($judul == false) {
        return $this->findAll();
    }
    return $this->like('judul', $judul)->findAll();
}
```

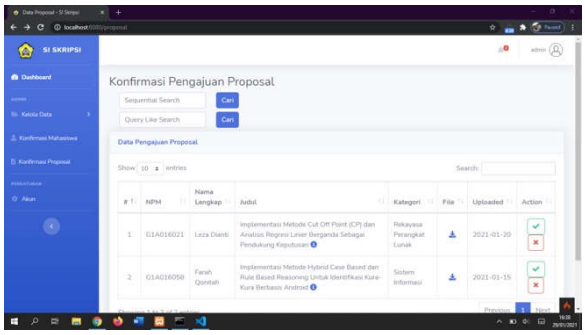
Gambar 2. Code pencarian dengan kondisi Like

Mekanisme yang dilakukan pada pencarian dengan kondisi Like adalah membaca semua *record* pada tabel judul skripsi kemudian menggantikannya dengan karakter sesuai dengan *keyword* yang di-*input*. Karakter yang tergantikan tersebut menunjukkan kecocokan *string,* kemudian proses dihentikan.

III. HASIL DAN PEMBAHASAN

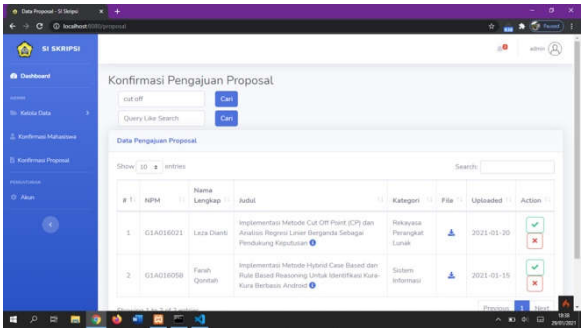
A. Implementasi pada Sistem

Setalah dilakukan pengembangan sistem informasi pengajuan judul skripsi, pada fitur pencarian peneliti membuat dua buah fungsi pencarian judul dengan dua metode *string matching*. Adapun halaman tersebut ditampilkan pada Gambar 3.

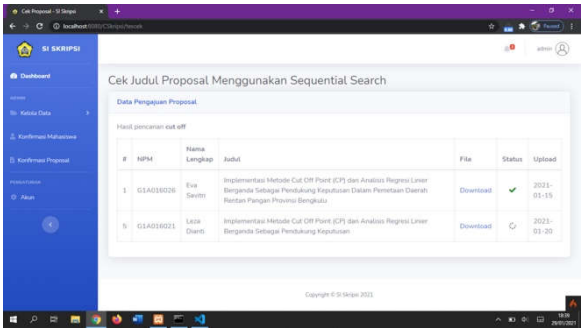


Gambar 3. Halaman pencarian judul skripsi

Pada halaman tersebut, belum dimasukkan *keyword* judul untuk menemukan judul skripsi. Gambar 4 dan 5 menunjukkan hasil pencarian dengan *keyword* yang sama melalui kolom pencarian dengan metode *sequential searching*.

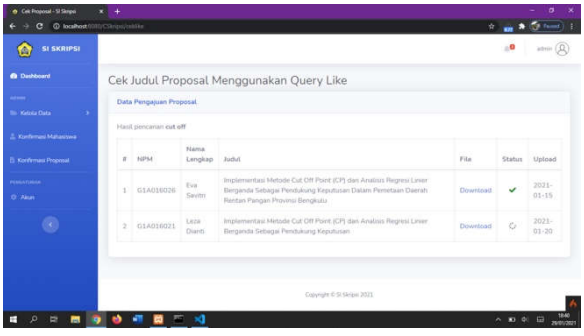


Gambar 4. Masukan keyword pada pencarian dengan sequential searching



Gambar 5. Hasil pencarian dengan sequential searching

Setelah melakukan pencarian menggunakan metode *sequential searching* dengan *keyword* yang sama, selanjutnya dilakukan dengan kondisi Like pada SQL. Adapun hasil yang diperoleh ditunjukkan pada Gambar 6.



Gambar 6. Hasil pencarian dengan kondisi Like

Berdasarkan percobaan yang telah dilakukan, kedua metode ini menunjukkan hasil yang sama. Metode pencarian menggunakan *sequential searching* dilakukan dengan memasukkan semua

data judul yang terdapat dalam *database* ke dalam *array*, kemudian dilakukan pencocokan *string* pada seluruh *record* sesuai dengan *keyword*. Sementara kondisi Like melakukan pencocokan *string* dengan membaca seluruh *record* pada tabel judul skripsi kemudian mengganti semua karakter sesuai dengan *keyword*. Dengan menggunakan data (*record*) judul skripsi yang sedikit kedua metode ini tidak menunjukkan perbedaan sama sekali, namun apabila dilakukan pencarian pada data yang lebih besar, tentu kondisi Like akan menunjukkan keefektifan yang lebih tinggi dibanding *sequential searching*. Menurut Susantha Bathige, kondisi Like dapat ditingkatkan keefektifannya hingga lebih dari 50% dalam melakukan pencocokan *string* jika menggunakan fungsi Reverse [7].

IV. KESIMPULAN DAN SARAN

Dari hasil analisa dan pembahasan yang telah dilakukan, maka dihasilkan kesimpulan:

1. Algoritme *exact string matching* dapat diterapkan pada fitur pencarian judul skripsi dalam sistem informasi pengajuan skripsi Program Studi Sistem Informasi Universitas Bengkulu.
2. *Sequential searching* dan kondisi Like pada sintaks SQL kurang efektif (terutama pada jumlah data yang besar) diterapkan pada fitur pencarian, karena kedua metode tersebut membaca seluruh *record* pada database berdasarkan *keyword* yang di-*input*-kan.

Saran yang dapat dilakukan pada penelitian selanjutnya adalah menambahkan fungsi Reverse pada kondisi Like dalam sintaks SQL agar proses pencocokan *string* dapat dilakukanlebih efektif.

DAFTAR PUSTAKA

[1] I. Markić, M. Štula, M. Zorić, and D. Stipaničev, “Entropy-Based Approach in Selection Exact String-Matching Algorithms,” *Entropy*, vol. 23, no. 1, p. 31, 2021.

[2] J. I. Sinaga, E. B. Mesran, and E. Buulo, “Aplikasi Mobile Pencarian Kata Pada Arti Ayat Al-Qur’an Berbasis Android Menggunakan Algoritme String Matching,” *INFOTEK*, vol. 2, pp. 68–72, 2016.

[3] V. Sagita and M. I. Prasetiyowati, “studi perbandingan implementasi algoritme boyer-moore, turbo boyer-moore, dan tuned boyer-moore dalam pencarian string,” *Ultim. J. Tek. Inform.*, vol. 5, no. 1, pp. 31–37, 2013.

[4] E. H. S. Atmaja and E. H. Parnadi, “Aplikasi Penjadwalan Perkuliahan Menggunakan Algoritme Sequential Search Dan Forward Checking,” *SEMNASTEKNOMEDIA ONLINE*, vol. 2, no. 1, pp. 2–4, 2014.

[5] G. Gunawan, “Aplikasi Kamus Istilah Ekonomi (Inggris-Indonesia) Menggunakan Metode Sequential Searching,” *Pseudocode*, vol. 3, no. 2, pp. 122–128, 2016.

[6] D. Lorentz and J. Gregoire, “SQL Reference,” in *B10759-01*, 10.1., Redwood CIty: Oracle Corporation, 2003, pp. 6–22.

[7] S. Bathige, “How to make SQL Server Wildcard Searches Faster,” 2018. .