Optimasi Query Pada Human Resource Information System (HRIS) di Universitas XYZ

Query Optimation on Human Resource Information System (HRIS) in Universitas XYZ

Hery Siswanto[1], Tri Andi[2], Kusrini[3]

[123]Magister Teknik Informatika, Universitas AMIKOM Yogyakarta
Jl. Ring Road Utara, Condong Catur, Depok, Sleman, Yogyakarta 55281
Email: hery505@gmail.com[1], anditri14@gmail.com[2], kusrini@amikom.ac.id[3]

ABSTRAK

Optimasi merupakan suatu langkah untuk mengoptimalkan waktu menjadi lebih efisien. Ketika sebuah query diberikan pada sistem database, optimasi penting dilakukan untuk memilih strategi yang efisien untuk mengevaluasi ekspresi relasi yang ditentukan. Query optimization adalah suatu proses untuk menganalisis query, menentukan sumber-sumber apa saja yang digunakan oleh query tersebut dan apakah penggunaan dari sumber tersebut dapat dikurangi tanpa merubah output. Kegiatan Pengelolaan manajemen sumber daya manusia yang baik sangat tergantung pada kualitas informasi untuk pengambilan keputusan dibidang sumber daya manusia. Kemampuan organisasi dalam memperoleh, menyimpan, memelihara dan menggunakan informasi sumber daya manusia merupakan faktor penting yang menunjang keberlangsungan hidup perusahaan. Perusahaan harus menyadari pentingnya pemenuhan kebutuhan sumber daya manusia secara berkualitas dan tepat sehingga perlu untuk dikembangkan sistem informasi sumber daya manusia untuk menunjang pemenuhan sumber daya manusia yang berkualitas, Sistem ini yang namanya biasa disebut SISDM atau HRIS (Human Resources Information System). Hasil yang diperoleh dari pengujian sebelum optimasi dan sesudah dioptimasi menunjukan bahwa query yang sudah di optimasi waktu yang di butuhkan dalam melakukan pencarian data lebih cepat. Hasil percobaan pertama dengan 1000 data waktu yang dibutuhkan sebelum optimasi 0.023 dan sesudah di optimasi waktu yang didapatkan 0.015.

Kata kunci: Human Resources Information System, data, Query

ABSTRACT

Optimization is a step to optimize time to be more efficient. When a query is given to a database system, optimization is important to choose an efficient strategy to evaluate the expression of the specified relation. Query optimization is a process for analyzing queries, determining what sources are used by the query and whether the use of these sources can be reduced without changing the output. Activities Management of good human resource management is highly dependent on the quality of information for decision making in the field of human resources. The ability of an organization to obtain, store, maintain and use information on human resources is an important factor that supports the survival of the company. Companies must realize the importance of meeting the needs of human resources in a quality and appropriate so that it needs to develop a human resource information system to support the fulfillment of quality human resources. This system, whose name is commonly called SISDM or HRIS (Human Resources Information System). The results obtained from testing before optimization and after optimization show that the queries that have been optimized for the time needed to perform data searches are faster. The first experimental results with 1000 time data needed before 0.023 optimization and after optimization time obtained 0.015.

Keywords: *Human Resources Information System, data, Query*

1. PENDAHULUAN

Kegiatan Pengelolaan manajemen sumber daya manusia yang baik sangat tergantung pada kualitas informasi untuk pengambilan keputusan dibidang sumber daya manusia. Kemampuan organisasi dalam memperoleh, menyimpan, memelihara dan menggunakan informasi sumber daya manusia merupakan faktor penting yang menunjang keberlangsungan hidup perusahaan. Perusahaan harus menyadari pentingnya pemenuhan kebutuhan sumber daya manusia

secara berkualitas dan tepat sehingga perlu untuk dikembangkan sistem informasi sumber daya manusia untuk menunjang pemenuhan sumber daya manusia yang berkualitas, Sistem ini yang namanya biasa disebut SISDM atau HRIS (Human Resources Information System) (Kavanagh dan Johnson, 2017).

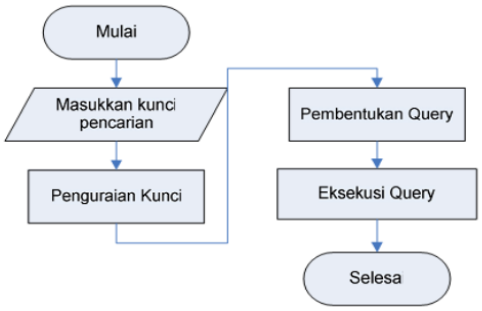
Dalam penelitiannya yang berjudul Optimasi Query Pada Sistem FAQ Di Suara Warga Universitas Negeri Semarang mengimplementasikan dan mengetahui model query dengan kinerja terbaik pada sistem FAQ di Suara Warga UNNES. Penelitian tersebut diujikan sebanyak 25 data Suara Warga UNNES dan 45 data dari inputan 9 responden mahasiswa UNNES pada 5 topik bahasan dengan 3 model query. Hasil pengujian yang dilakukan nilai recall 0.92, precision 0.41, fmeasure 0.56 dengan model query isi, nilai recall 0.86, precision 0.59, fmeasure 0.70 dengan model query isi dengan feature selection, nilai recall 0.88, precision 0.78, fmeasure 0.82 dengan model query judul. Dari hasil tersebut model query dari judul inputan memiliki nilai paling tinggi dari ke-3 pengujian tersebut (Tama, 2015).

Dalam penelitianya Optimasi Query Untuk Pencarian Data Menggunakan Penguraian Kalimat penelitian tersebut menggunakan 3 skenario uji coba dimana setiap skenario mengambil 5 contoh pencarian dengan beberapa kata acak dari judul yang hendak dicari. Hasilnya sebelum dilakukan optimasi indetifikasi personal membutuhkan waktu 2 dan setelah diptimasi 26 (Sanjaya, 2016). Dari hasil penelitian sebelumnya oleh karena itu penulis disini ingin melalkukan penelitian Optimasi Query Pada Human Resource Information System (HRIS) di Universitas XYZ.

2. PEMBAHASAN

2.1 Alur

Alur optimasi query untuk pencarian data menggunakan penguraian kalimat dimulai dari input kunci pencarian. Kunci pencarian bisa berupa susunan 2 kata atau bisa berupa kalimat. Kemudian inputan kunci pencarian diurai menjadi kata yang terpisah. Tahap selanjutnya yaitu penyusunan query. Terakhir query yang sudah tersusun dieksekusi (Conoras dan Kurnawan, 2016). Alur optimasi query disajikan pada Gambar 1.



Gambar 1. Menunjukan Alur Query

2.2 Analisis

Studi kasus yang diambil sebagai contoh uji coba adalah database yang ada di sistem informasi kepegawain yang ada di Universitas XYZ. Sistem Sistem Informasi Manajemen Kepegawaian merupakan suatu sistem informasi manajemen yang berfungsi untuk mengelola data, manajemen dan administrasi kepegawaian sebuah instansi, perguruan tinggi ataupun perusahaaan. HRIS menjadi solusi tepat bagi sebuah instansi, perusahaaan ataupun perguruan tinggi dalam mengatasi masalah manajemen kepegawaian.

Tujuan Implementasi HRIS adalah dapat terwujudnya suatu sistem informasi manajemen yang berintegrasi dalam suatu jaringan komputer yang mampu menghasilkan informasi yang bermutu untuk menunjang pengambilan keputusan manajemen kepegawaian di lingkungan instansi. Kegiatan ini juga mendukung proses bisnis serta kelangsungan sebuah instansi, perusahaan, ataupun perguruan tinggi. Oleh karena itu, komitmen sebuah instansi untuk menjalankan Sistem Informasi Manajemen haruslah sangat tinggi agar proses yang terjadi dilantai produksi menjadi menguntungkan bagi pengguna (Marimin, Tanjung dan Prabowo, 2006).

2.3 Optimalisasi

Optimasi Query adalah suatu proses untuk menganalisa query untuk menentukan sumber-sumber apa saja yang digunakan oleh query tersebut dan apakah penggunaan dari sumber tersebut dapat dikurangi tanpa merubah output. Atau bisa juga dikatakan bahwa optimasi query adalah sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu query untuk membuat evaluasi tersebut menjadi lebih efektif. Optimasi query mencakup beberapa teknik seperti transformasi query ke dalam bentuk logika yang sama, memilih jalan akses yang optimal dan mengoptimumkan penyimpanan data.

Tujuan dari optimasi query adalah menemukan jalan akses yang termurah untuk

meminimumkan total waktu pada saat proses sebuah query. Untuk mencapai tujuan tersebut, maka diperlukan optimizer untuk melakukan analisa query dan untuk melakukan pencarian jalan akses.

Beberapa cara optimasi query SQL adalah sebagai berikut:

1. Index

Mengindeks kolom dalam tabel adalah cara yang biasa dilakukan untuk mengoptimalkan hasil pencarian query SQL. Membuat indeks dalam sql tak ubahnya seperti membuat daftar indeks pada buku. Bayangkan ketika anda ingin mencari sebuah section pada buku mengenai suatu hal dan harus mencari setiap halaman dari awal sampai akhir, dengan indeks anda cukup mencari urutan indeks kata tersebut dan langsung membuka halaman rujukan yang ada pada indeks. Penggunaan indeks pada eksekusi query SQL akan benar-benar terasa ketika tabel dan basis data yang kita punya sudah cukup besar.

Apakah index scan selalu lebih cepat dibandingkan dengan table scan? Ternyata tidak juga! Table scan bisa jadi bekerja lebih cepat saat mengakses record dalam jumlah relatif kecil, ataupun pada saat aplikasi memang memerlukan pembacaan table secara keseluruhan. Sebaliknya dalam mengakses record yang besar pada field tertentu, index scan dapat mengurangi operasi pembacaan I/O sehingga tidak jarang menghasilkan kinerja yang lebih cepat.

2. Symbol Operator

Penggunaan operator simbol seperti <, >, =, !=, ><, dll sangat membantu dalam menghasilkan kecepatan pencarian dari *query*. Cara kerja *database management system* (DBMS) dalam menerapkan query dengan kondisi “>” dibandingkan dengan “=>” akan sedikit berbeda, apabila kita menggunakan “=>” maka DBMS akan menerapkan dua kriteria pada setiap hasil, tidak langsung mencari pada kondisi langsung seperti apabila kita menggunakan “>”.

3. *Wildcard* (kartu liar)

Wildcard yang digambarkan dengan simbol “%” yang kita tuliskan pada kondisi/kriteria *query* akan sangat berpengaruh pada lamanya proses pencarian. *Wildcard* sendiri dapat kita bagi menjadi tiga bagian penggunaan, yaitu *wildcard* penuh (contoh ‘%kata%’), *postfix wildcard* (contoh ‘kata%’), dan *prefix wildcard* (contoh ‘%kata’). Sebisa mungkin hindari penggunaan *wildcard* penuh karena untuk ukuran tabel dengan baris yang banyak, dan pencarian kolom dengan panjang kolom yang besar akan benar-benar menyiksa basis data dalam melakukan pencarian.

4. Operator Negatif

Penggunaan operator negatif (seperti NOT, NOT LIKE, NOT IN, NOT EXIST, != dll) akan lebih memakan waktu lama dibandingkan operator positif. Operator = merupakan operator paling memudahkan database dalam melakukan pencarian, terutama ketika database sudah terindeks maka pencarian *exact match* ini akan dihasilkan dengan cepat.

5. COUNT vs EXIST

Seringkali dalam pembuatan sebuah *loop* dan logika dalam aplikasi- kita ingin membuat penguji sederhana dengan pembuatan perintah yang dilaksanakan hanya jika sebuah nilai ada dalam tabel (pada kolom tertentu dalam tabel) – kita menggunakan COUNT sebagai kriteria. Dalam artian ketika COUNT menghasilkan angka > 0 maka loop atau logika akan dijalankan.

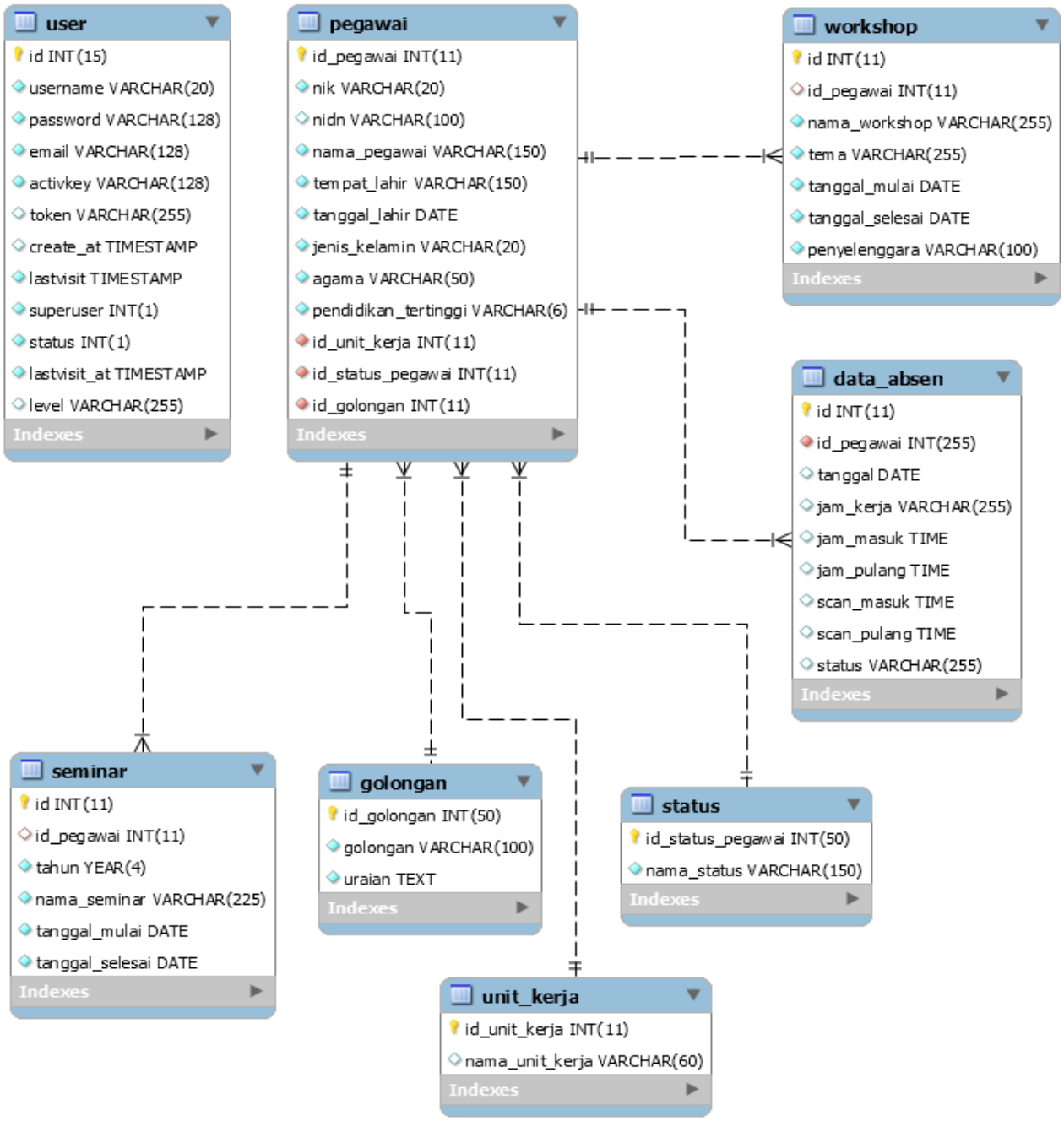
```
SELECT KOLOM FROM TABEL WHERE COUNT > 0
```

Penggunaan COUNT akan melakukan pencarian dari awal hingga akhir baris dari tabel. Sebagai alternatif yang lebih baik anda dapat menggunakan EXIST, kriteria ini akan menghentikan pencarian ketika kriteria/kondisi sudah ditemukan.

Sebagai contoh untuk menentukan index pada field yang sering digunakan, misalnya field yang sering diakses oleh klausa WHERE, JOIN, ORDER BY, GROUP BY.

Dari beberapa cara tersebut hanya sebagian saja yang akan diterapakan dalam optimasi HRIS yang dibahas dalam penelitian ini.

Efektifitas dalam pencarian data sangat mutlak diperlukan. Salah satu permasalahan yang dihadapi penulis dalam hal pencarian data pada database yaitu ketika mencari suatu data dengan kata kunci sebanyak 2 atau lebih (berupa kalimat) serta susunannya terbalik, hasil dari pencarian tersebut menjadi terbatas dan tidak menemukan hasil. Penelitian ini mencoba melakukan optimasi query untuk pencarian data yaitu dengan menguraikan kalimat yang dijadikan acuan pencarian menjadi kata kunci dan di kombinasikan dengan operator OR pada syarat pencarian dan hash join untuk menggabungkan data-data yang berjumlah besar (Conoras dan Kurnawan, 2016).



Gambar 2. Basis Data HRIS

3. PERCOBAAN PERTAMA

Sebagai uji coba penelitian, tabel jenis yang akan di uji jumlah data dan tabel yang menggunakan paritisi dengan Skala jumlah data yang terlihat pada tabel, hal ini dilakukan untuk memudahkan uji eksekusi query dengan jumlah data yang berbeda-beda (Sulhan dan Anshori, 2016). Dalam percobaan pertama ini Menampilkan data pegawai Nik, nama pegawai, golongan, unit kerja status, tanggal lahir dan umur. Data pegawai ditunjukan pada Gambar 2.

```
+----------+--------------+----------+----------------------+-------------+---------------+---
---+
| nik      | nama_pegawai | golongan | nama_unit_kerja      | nama_status | tanggal_lahir |
usia |
+----------+--------------+----------+----------------------+-------------+---------------+---
---+
| 2017074  | Pegawai 74   | III/B    | S2 Psikologi Profesi | DOSEN TETAP | 1970-12-10    |
47 |
```

```
| 20170116 | Pegawai 116  | III/B    | S2 Psikologi Profesi | DOSEN TETAP | 1970-12-10    |
47 |
| 20170122 | Pegawai 122  | IV/D     | S2 Psikologi Profesi | DOSEN TETAP | 1970-12-10    |
47 |
| 20170212 | Pegawai 212  | I/A      | S2 Psikologi Profesi | DOSEN TETAP | 1980-12-09    |
37 |
| 20170254 | Pegawai 254  | I/A      | S2 Psikologi Profesi | DOSEN TETAP | 1980-12-09    |
37 |
| 20170260 | Pegawai 260  | II/C     | S2 Psikologi Profesi | DOSEN TETAP | 1980-12-09    |
37 |
| 20170350 | Pegawai 350  | IV/A     | S2 Psikologi Profesi | DOSEN TETAP | 1980-12-09    |
37 |
| 20170392 | Pegawai 392  | IV/A     | S2 Psikologi Profesi | DOSEN TETAP | 1980-12-09    |
37 |
| 20170398 | Pegawai 398  | IV/D     | S2 Psikologi Profesi | DOSEN TETAP | 1980-12-09    |
37 |
| 20170488 | Pegawai 488  | I/D      | S2 Psikologi Profesi | DOSEN TETAP | 1980-12-09    |
37 |
+----------+--------------+----------+----------------------+-------------+---------------+---
---+
10 rows in set
```

Gambar 3. Tampilan data pegawai

Perintah Select pegawai digunakan untuk memilih kolom dengan nama pegawai yang berisi data pegawai, untuk menggabungkan syarat pencarian berdasarkan hasil penguraian data. Adapun perintah untuk menyusun query sebelum optimasi dan sesudah optimasi adalah sebagai berikut :

Sebelum optimasi

```
SELECT
pegawai.nik,
pegawai.nama_pegawai,
golongan.golongan,
unit_kerja.nama_unit_kerja,
`status`.nama_status, tanggal_lahir,
YEAR(CURDATE())                    -
YEAR(tanggal_lahir) AS usia
FROM
golongan, unit_kerja, status, pegawai
where
golongan.id_golongan=
pegawai.id_golongan AND
unit_kerja.id_unit_kerja           =
pegawai.id_unit_kerja AND
pegawai.id_status_pegawai          =
`status`.id_status_pegawai;
```

Dengan optimasi

```
SELECT
pegawai.nik,
pegawai.nama_pegawai,
golongan.golongan,
unit_kerja.nama_unit_kerja,
`status`.nama_status, tanggal_lahir,
YEAR(CURDATE())                    -
YEAR(tanggal_lahir) AS usia
FROM
pegawai
INNER     JOIN     golongan     ON
pegawai.id_golongan                =
golongan.id_golongan
INNER     JOIN     unit_kerja     ON
pegawai.id_unit_kerja              =
unit_kerja.id_unit_kerja
INNER     JOIN     `status`     ON
pegawai.id_status_pegawai          =
`status`.id_status_pegawai;
```

Pada Tabel 1 menunjukan hasil pengujian pertama ini dihasilkan sebelum menggunakan Optimasi dan Setelah optimasi.

Tabel 1. Hasil sebelum dan sesudah optimasi

Jumlah data	Sebelum optimasi (s)	Sesudah Optimasi (s)
1000	0.023	0.015
2000	0.031	0.028
4000	0.058	0.055
8000	0.118	0.111

4. PERCOBAAN KEDUA

Mencari Rentang Umur Perintah query untuk menghitung jumlah pegawai berdasarkan range umur. Rentang umur ditunjukan pada Tabel 2.

Tabel 2. Rentang Umur

NO	Range Umur
1	<30
2	31 - 40
3	41 - 51
4	51 - 60
5	>61

Sebelum optimasi

```
SELECT
    CASE
        WHEN umur < 30 THEN '... -
30'
        WHEN umur BETWEEN 31 and 40
THEN '31 - 40'
        WHEN umur BETWEEN 41 and 51
THEN '41 - 51'
        WHEN umur BETWEEN 51 and 60
```

```
THEN '51 - 60'
        WHEN umur >= 61 THEN '61 -
...'
    END as range_umur,
    COUNT(*) AS jumlah
FROM      (select      nama_pegawai,
tanggal_lahir,   TIMESTAMPDIFF(YEAR,
tanggal_lahir,  CURDATE())  AS  umur
from pegawai)  as dummy_table
GROUP BY range_umur
ORDER BY range_umur;
```

Sesudah optimasi

```
SELECT
 COUNT(IF(umur < 30,1,NULL)) AS '...
- 30',
 COUNT(IF(umur   BETWEEN   31   and
40,1,NULL)) AS '31 - 40',
 COUNT(IF(umur   BETWEEN   41   and
50,1,NULL)) AS '41 - 50',
 COUNT(IF(umur   BETWEEN   51   and
60,1,NULL)) AS '51 - 60',
 COUNT(IF(umur >= 61,1,NULL)) AS '61
- ...'
FROM      (select      nama_pegawai,
tanggal_lahir,   TIMESTAMPDIFF(YEAR,
tanggal_lahir,  CURDATE())  AS  umur
from pegawai) as dummy_table;
```

Dari hasil percobaan kedua hasil setelah optimasi sama bahwa waktu yang dibutuhkan dalam pencarian data lebih cepat dibandingkan sebelum optimasi, hasil dari percobaan kedua ini ditunjukan pada Tabel 3.

Tabel 3. Hasil percobaan kedua

Jumlah data	Sebelum optimasi (s)	Sesudah Optimasi (s)
1000	0.007	0.005
2000	0.012	0.009
4000	0.022	0.014
8000	0.042	0.029

5. KESIMPULAN

Berdasarkan hasil uji query dan analisa partisi tabel pada pengambilan data di Sistem Pegawai Universitas XYZ, maka dapat diambil kesimpulan bahwa pengujian yang dilakukan untuk hasil perbedaan query yang di akses di tabel belum dioptimasi dan tabel yang dioptimasi waktu yang paling cepat adalah dari tabel yang dioptimasi terbukti Untuk tabel yang di optimasi waktu yang di perlukan 0.015 sedangkan waktu yang diperlukan untuk akses data pada tabel yang tidak dioptimasi 0.023. Percobaan pertama mengunakan query biasa dan mengunakan joint. Percobaan kedua mencarai rentang umur mengunakan case dan if pada query dari hasil percubaan mengunakan if waktu untuk mencari data lebih cepat.

DAFTAR PUSTAKA

A. Sanjaya, “Optimasi Query Untuk Pencarian Data Menggunakan Penguraian Kalimat,” no. November, pp. 6–7, 2016.

G. A. Tama, “Optimasi Model Query Pada Sistem Faq Di Suara Warga Universitas Negeri Semarang,” 2015.

Marimin, Hendri Tanjung, and H. Prabowo, Sistem Informasi Manajemen Sumber Daya Manusia. Grasindo.

M. E. B. Conoras and A. D. Kurnawan, “Optimasi Query Untuk Pencarian Data Menggunakan Penguraian Kalimat,” no. November, pp. 6–7, 2016.

M. J. Kavanagh and R. D. Johnson, *Human Resource Information Systems: Basics, Applications, and Future Directions*. SAGE Publications, 2017.

M. Sulhan and I. Anshori, “Menggunakan Tabel Terpartisi dan Tabel Tidak Terpastisi dengan Metode Cost-Based,” SMATIKA J., vol. 6, no. 2, pp. 19–23, 2016.