# PERBANDINGAN PERFORMA OPTIMASI QUERY SCALAR, CORRELATED DAN KOMBINASI PADA ORACLE

*COMPARISON OF SCALAR, CORRELATED AND COMBINATION QUERY OPTIMIZATION PERFORMANCE ON ORACLE*

**Sinta Ayu Rismawati[1*], Nasywa Zahira Ramadhani[2], Elisa Tri Aswuri[3], Siti Mukaromah[4]**
*E-mail: sintaayr03@gmail.com

[1,2,3,4]Sistem Informasi, Fakultas Ilmu Komputer, UPN "Veteran" Jawa Timur

**Abstrak**

Di era informasi yang terus berkembang, DBMS sangat penting untuk pengelolaan dan penyimpanan informasi yang efektif. Salah satu DBMS yang populer yaitu Oracle. Saat menggunakan Oracle, penyajian dan permintaan informasi yang efisien sangat penting, sehingga perlu dipastikan apakah query yang dipakai dalam menjalankan perintah bekerja secara optimal dan memberikan hasil yang diinginkan dengan cepat. Pada artikel ini dilakukan perbandingan elapsed time dari tiga jenis query yaitu *scalar, correlated* dan kombinasi terhadap sejumlah data beserta beberapa relasi. Tujuan artikel ini adalah untuk mengetahui query mana yang paling efektif diantara ketiga query tersebut untuk digunakan di lingkungan *oracle*. Data yang digunakan adalah dari skema hr pada database oracle. Pengujian dilakukan pada jumlah baris data 20, 40, 60, 80, dan 100. Hasil perbandingan menunjukkan bahwa jenis *query correlated* lebih optimal untuk jumlah relasi yang sedikit. Sedangkan jenis q*uery cross-product scalar* lebih optimal dibandingkan query lainnya untuk jumlah data dan jumlah relasi yang banyak.
**Kata kunci :** *oracle, scalar, correlated, kombinasi, optimal, elapsed time*

**Abstract**

*In the ever-evolving information age, DBMS is essential for the effective management and storage of information. One of the popular DBMS is Oracle. When using Oracle, efficient serving and querying of information is essential. So it is necessary to ensure whether the query used to execute the command works optimally and gives the desired results quickly. In this article, a comparison of the elapsed time of three types of queries, namely scalar, correlated and combined, is carried out on a number of data along with several relations. The purpose of this article is to find out which query is the most effective among the three queries for use in the oracle environment. The data used is from the hr schema on the oracle database. Tests were carried out on the number of data lines 20, 40, 60, 80, and 100. The comparison results show that the correlated query type is more optimal for a small number of relationships. While the type of scalar cross product query is more optimal than other queries for large amounts of data and relationships.*
***Keywords:*** *oracle, scalar, correlated, combination, optimal, elapsed time*

## 1. PENDAHULUAN

Di era informasi yang terus berkembang, sistem manajemen basis data (DBMS) sangat penting untuk pengelolaan dan penyimpanan informasi yang efektif. DBMS adalah software yang dibuat untuk memudahkan proses input, edit dan hapus serta menampilkan data [1]. DBMS dapat melakukan pengolahan data yang berskala besar maupun kecil dengan performa pemilihan database yang terjaga dengan baik [2]. Faktor-faktor yang mempengaruhi performa DBMS antara lain yaitu waktu respon, *throughput*, *resources* dan *memory* [3].

Database merupakan kumpulan data yang saling berelasi serta dapat digunakan untuk mendapatkan informasi yang dibutuhkan [4]. Salah satu sistem DBMS yang populer adalah Oracle, yang banyak digunakan dalam berbagai aplikasi dan bisnis. Salah satu kemampuan Oracle yang dapat dimanfaatkan oleh seorang database engineer yaitu pembuatan query atau perintah untuk menampilkan data-data yang ada pada tabel-tabel di dalam database [5]. Namun, sebuah query mungkin memiliki biaya eksekusi yang mahal jika tidak dieksekusikan dengan baik. Hal tersebut bisa berdampak negatif bagi perusahaan karena kinerja performa yang menurun [6].
*Query* adalah perintah yang digunakan untuk mengambil data dalam sebuah DBMS, termasuk pada Oracle [7]. Sebagai pengguna Oracle, perlu untuk memastikan bahwa *query* yang diinputkan tersebut optimal dan memberikan hasil yang diinginkan dengan cepat. Optimasi query bertujuan untuk meningkatkan proses akses data menjadi lebih efektif. Dalam optimasi query terdapat beberapa teknik seperti melakukan transformasi pada query ke dalam bentuk query baru dengan logika yang sama, mengoptimalkan penyimpanan data dan memilih akses data yang tercepat [8]. Beberapa teknik pengoptimalan dapat digunakan untuk meningkatkan kinerja dan efisiensi query pada Oracle. Dalam artikel ini, kami fokus pada tiga teknik optimasi query, yaitu *query scalar*, *query correlated* dan *query* kombinasi. Pertama, *query scalar* adalah metode pengoptimalan yang menggunakan operator *scalar* untuk memproses *query* secara efisien. Kedua, *query correlated* adalah metode yang menghubungkan subquery ke query induk melalui kondisi terkait. Dalam metode ini, sebuah sub query dijalankan untuk setiap baris dari query utama yang dapat menghasilkan hasil yang lebih akurat dan efisien. Ketiga, metode optimasi *query* kombinasi dengan menggabungkan kekuatan *query scalar* dan *query correlated*. Dalam metode ini dapat menggunakan operator scalar untuk memproses query secara lebih efisien serta menjaga hubungan antar sub query dan query utama untuk mendapatkan hasil yang akurat.
Tujuan dari artikel ini adalah untuk membandingkan kinerja dari tiga metode optimasi, yaitu *query scalar*, *query correlated* dan *query* kombinasi, ditinjau dari waktu eksekusi query dan pemanfaatan sumber daya di Oracle. Dengan membandingkan hasil dari ketiga metode tersebut, kita dapat menentukan metode mana yang paling efektif untuk optimasi query di lingkungan Oracle. Selain itu, penting untuk memastikan bahwa pada penulisan artikel ini, data dan informasi yang digunakan dari sumber lain dikutip dengan jelas dan dimasukkan dalam daftar referensi yang sesuai. Melalui artikel ini, diharapkan dapat memberikan informasi kepada pengguna Oracle untuk memilih metode pengoptimalan query yang tepat, dengan tujuan meningkatkan kinerja dan efisiensi pengambilan data di lingkungan Oracle.

## 2. METODOLOGI

Tahapan penelitian digambarkan pada Gambar 1.


**Gambar 1. Alur kerangka kerja**

### 2.1 Studi Literatur

***Query Scalar*** adalah subset query yang hanya mengembalikan satu nilai (baris dan kolom) dengan melakukan *select* beberapa kolom dari sebuah tabel dimana terdapat kondisi tertentu pada kolom yang terpenuhi pada sebuah sub query [9]. Sintaks query scalar dijelaskan pada Gambar 2.

```
select[nama_kolom1],...,[nama_kolomN]
from [nama_tabel1]
where [nama_tabel1].[nama_kolom1] in
(select[nama_kolom1] from [nama_tabel2])
where [nama_tabel1].[nama_kolom1] in
(select[nama_kolom1] from [nama_tabel2])
```

**Gambar 2. Scalar**

***Query Correlated*** adalah subset query yang bergantung pada hasil outer query dengan melakukan *select* beberapa kolom dari sebuah tabel dimana dalam kondisi sebuah kolom terpenuhi pada sebuah sub query, dan sub query tersebut masih terkait dengan super query atau outer query [9]. Sintaks query correlated dijelaskan pada Gambar 3.

```
select[nama_kolom1],...,[nama_kolomN]
from [nama_tabel1]
where [nama_tabel1].[nama_kolom1] in
(select[nama_kolom1] from [nama_tabel2])
where [nama_tabel1].[nama_kolom2] = [nama_tabel2].[nama_kolom2])
```

**Gambar 3. Correlated**

***Query Kombinasi*** adalah hasil penggabungan subset query dengan bentuk *cross product* [9]. Kombinasi query tersebut dibagi menjadi dua jenis :

1. *Cross Product* dengan *Scala*
   Pada query cross product dengan scalar, subset query melakukan *select* kolom tertentu dari beberapa tabel, dimana dalam kondisi antar tabel tersebut menggunakan operasi join dan sebuah kolom terpenuhi oleh sebuah sub query. Sintaks query kombinasi ada pada Gambar 4.

```
select [nama_kolom1],...,[nama_kolomN]
from [nama_tabel1],[nama_tabel2]
where [nama_tabel1].[nama_kolom1] = [nama_tabel2].[nama_kolom2]
AND [nama_tabel1].[nama_kolumn1] in
(select [nama_kolom1] from [nama_tabel3])
```

**Gambar 4. Cross product dengan scalar**

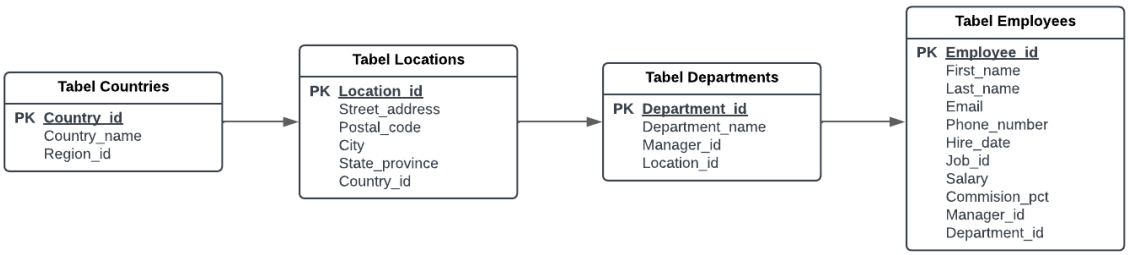
2. *Multi Scalar*
   Pada query multi scalar, subset query melakukan *select* kolom tertentu dari sebuah tabel, dimana dalam kondisi sebuah kolom terpenuhi menggunakan sebuah subquery, dan subquery tersebut memiliki sebuah sub query lainnya. Sintaks query ini dijelaskan pada gambar 5.

```
select [nama_kolom1],..., [nama_kolomN]
from [nama_tabel1],[nama_tabel2]
where [nama_tabel1].[nama_kolom1] in
(select [nama_kolom1] from [nama_tabel2]
where [nama_tabel2].[nama_kolom2] in
(select [nama_kolom2] from [nama_tabel3]))
```

**Gambar 5. Multi Scalar**

## 2.2 Pemilihan Dataset

Pada artikel ini, basis data yang digunakan adalah basis data oracle dengan skema hr. Pengujian query akan dilakukan pada 4 (empat) tabel yang saling berelasi. Adapun tabel yang saling berelasi tersebut akan ditampilkan pada Gambar 6.




**Gambar 6. Relasi antar tabel pengujian**

Masing - masing tabel memiliki jumlah baris data yang berbeda. Spesifikasi skema hr yang akan digunakan untuk pengujian dijelaskan pada Tabel 1.

**Tabel 1. Spesifikasi skema hr**

| No. | Nama Tabel | Jumlah Baris | Jumlah Kolom |
|---|---|---|---|
| 1. | Employees | 107 | 11 |
| 2. | Departments | 27 | 4 |
| 3. | Locations | 23 | 6 |
| 4. | Countries | 25 | 3 |

## 2.3 Implementasi dan Pengujian Query

Pengujian query pada artikel ini dilakukan pada tiga jenis subquery yang berbeda, yaitu *query scalar*, *query correlated*, dan *query kombinasi*. Terdapat lima kategori jumlah data yang digunakan untuk pengujian, mulai dari 20, 40, 60, 80, hingga 100 data (Tabel 2). Pada setiap jumlah data dilakukan pengujian sebanyak tiga kali [10]. Pencarian data pada query dibagi berdasarkan jumlah relasi antar tabel yang nantinya akan menghasilkan data yang sama. Relasi yang digunakan adalah relasi antar 2, 3, dan 4 tabel. Hasil pengujian performa query selanjutnya akan dilihat berdasarkan waktu kecepatan pemrosesan melalui *elapsed time*.

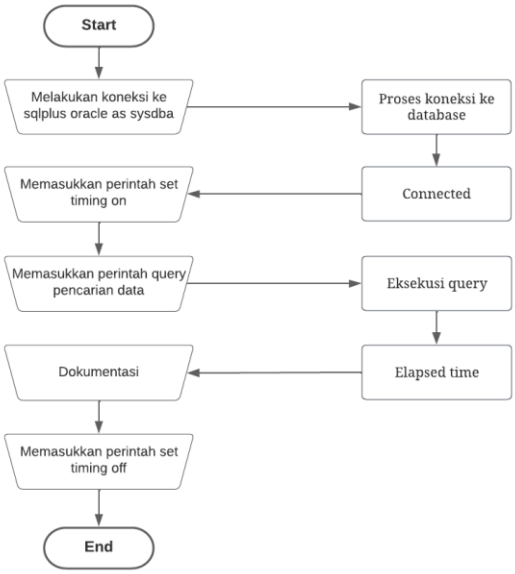
**Tabel 2. Jumlah data**

| No. | Pengujian ke- | Jumlah Data |
|---|---|---|
| 1. | 1 | 20 |
| 2. | 2 | 40 |
| 3. | 3 | 60 |
| 4. | 4 | 80 |
| 5. | 5 | 100 |

Untuk skenario pengujian query lebih jelasnya, akan ditampilkan pada flowchart di Tabel 3.

**Tabel 3. Flowchart skenario pengujian query**

| Operator | Sistem |
|---|---|




### 2.4 Analisis Hasil Query

Setelah tahap pengujian query secara keseluruhan, hasilnya akan dianalisis secara kuantitatif deskriptif untuk membandingkan performa masing - masing jenis query yang akan dibahas pada bab hasil dan pembahasan.

## 3. HASIL DAN PEMBAHASAN

Pada artikel ini, pengujian terhadap tiap jenis query dilakukan menggunkan sintaks query seperti pada Tabel 4, Tabel 5, Tabel 6, Tabel 7 dan Tabel 8. Pengujian dilakukan menggunakan Oracle 21c dan laptop Asus X415 sebagai perangkat utama yang dilengkapi dengan processor 11th Gen Intel Core i5 serta Ram 8GB.

### 3.1 Query Yang Digunakan

Sintaks untuk pengujian masing-masing jenis query disajikan pada Tabel 4.

**Tabel 4. Query scalar dan correlated pada 2 tabel sebanyak n baris**

| Scalar | Correlated |
|---|---|
| select*from (select employee_id, first_name from employees where employees.department_id in ( select department_id from departments) ) where rownum <= n; | select*from( select employee_id,first_name from employees where employees.department_id in (select department_id from departments where employees.department_id= departments.department_id) ) where rownum <= n; |

Sintaks query pada Tabel 4 digunakan untuk pengujian jenis query scalar dan correlated pada 2 tabel yaitu tabel employees dan tabel department dengan jumlah baris tertentu.

**Tabel 5. Query scalar dan correlated pada 3 tabel sebanyak n baris**

| Scalar | Correlated |
|---|---|
| select*from (<br>select employee_id, first_name<br>from employees, departments<br>where employees.department_id in<br>(select department_id from<br>departments)<br>and departments.location_id in<br>(select location_id from locations)<br>)where rownum <= n; | select*from( select employee_id, first_name<br>from employees, departments<br>where employees.department_id in (<br>select department_id from departments where<br>employees.department_id=departments.department_id) and departments.location_id in (<br>select location_id from locations where<br>departments.location_id=locations.location_id)<br>)where rownum <= n; |

Sintaks query pada Tabel 5 digunakan untuk pengujian jenis query scalar dan correlated pada 3 tabel yaitu tabel employees, tabel department dan tabel locations dengan jumlah baris tertentu.

**Tabel 6. Query cross-product scalar dan multi scalar pada 3 tabel sebanyak n baris**

| Cross-Product dengan Scalar | Multi Scalar |
|---|---|
| select*from(select employee_id, first_name<br>from employees, departments where<br>employees.department_id=<br>departments.department_id and<br>departments.location_id in (<br>select location_id from locations)<br>)where rownum <= n; | select*from(select employee_id,<br>first_name from employees, departments<br>where employees.department_id in<br>(select department_id from departments<br>where departments.location_id in<br>(select location_id from locations))<br>)where rownum <= n; |

Sintaks query pada Tabel 6 digunakan untuk pengujian jenis query cross-product dengan scalar dan multiscalar pada 3 tabel yaitu tabel employees, tabel department dan tabel locations dengan jumlah baris tertentu.

**Tabel 7. Query scalar dan correlated pada 4 tabel sebanyak n baris**

| Scalar | Correlated |
|---|---|
| select employee_id, first_name from (<br>select employee_id, first_name from<br>employees, departments, locations where<br>employees.department_id in (<br>select department_id from departments)<br>and departments.location_id in (<br>select location_id from locations) and<br>locations.country_id in (<br>select country_id from countries)<br>)where rownum <= n; | select*from(select employee_id, first_name<br>from employees, departments, locations where<br>employees.department_id in<br>(select department_id from departments where<br>employees.department_id=departments.department_id) and epartments.location_id in<br>(select location_id from locations where<br>departments.location_id=locations.location_id) and locations.country_id in (select<br>country_id from countries where<br>locations.country_id=countries.country_id)<br>)where rownum <= n; |

Sintaks query pada Tabel 7 digunakan untuk pengujian jenis query scalar dan correlated pada 4 tabel yaitu tabel employees, tabel department, tabel locations dan tabel countries dengan jumlah baris tertentu.

**Tabel 8. Query cross-product scalar dan multi scalar pada 4 tabel sebanyak n baris**

| Cross-Product Dengan Scalar | Multi Scalar |
|---|---|

| | |
|---|---|
| select*from(select employee_id, first_name from employees, departments, locations where employees.department_id= departments.department_id and departments.location_id in ( select location_id from locations) and locations.country_id in ( select country_id from countries) )where rownum <= n; | select*from(select employee_id, first_name from employees, departments, locations where employees.department_id in((select department_id from departments where departments.location_id in (select location_id from locations where locations.country_id in (select country_id from countries)))) )where rownum <= n; |

Sintaks query pada Tabel 8 digunakan untuk pengujian jenis query cross-product dengan scalar dan multiscalar pada 3 tabel yaitu tabel employees, tabel department, tabel locations dan tabel countries dengan jumlah baris tertentu.

### 3.2 Hasil Perbandingan

Dari hasil pengujian query yang digunakan pada relasi dua hingga empat tabel didapatkan hasil perbandingan dari empat jenis query yang disajikan pada Tabel 9.

**Tabel 9. Perbandingan query pada 2 tabel**

| **Query** | **Jumlah Data (baris)** | | | | |
|---|---|---|---|---|---|
| | **20** | **40** | **60** | **80** | **100** |
| | **Elapsed Time (ms)** | | | | |
| **Scalar** | 5 | 7 | 8 | 11 | 17 |
| | 3 | 6 | 8 | 11 | 13 |
| | 3 | 4 | 8 | 10 | 13 |
| **Rata-Rata** | **3.7** | **5.7** | **8** | **10.7** | **14.3** |
| **Correlated** | 5 | 6 | 6 | 11 | 13 |
| | 3 | 5 | 8 | 11 | 14 |
| | 2 | 5 | 2 | 11 | 13 |
| **Rata-Rata** | **3.3** | **5.3** | **5.3** | **11** | **13.3** |
| **Tercepat** | Correlated | Correlated | Correlated | Scalar | Correlated |
| **Terlama** | Scalar | Scalar | Scalar | Correlated | Scalar |

Dalam pengujian relasi antara dua tabel hanya dapat menggunakan jenis *query scalar* dan *correlated*. Pada tabel 9, jenis *query correlated* memiliki rata-rata *elapsed time* yang lebih cepat dibandingkan rata-rata *elapsed time* dari *query scalar*.

**Tabel 10. Perbandingan query pada 3 tabel**

| **Query** | **Jumlah Data (baris)** | | | | |
|---|---|---|---|---|---|
| | **20** | **40** | **60** | **80** | **100** |
| | **Elapsed Time (ms)** | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **Scalar** | 1 | 4 | 4 | 7 | 11 |
| | 1 | 3 | 7 | 10 | 9 |
| | 1 | 3 | 4 | 6 | 11 |
| **Rata-Rata** | **1** | **3.3** | **5** | **7.7** | **10.3** |
| **Correlated** | 2 | 4 | 8 | 10 | 10 |
| | 1 | 3 | 7 | 10 | 11 |
| | 2 | 3 | 8 | 11 | 9 |
| **Rata-Rata** | **1.7** | **3.3** | **7.7** | **10.3** | **10** |
| **Cross-Product Scalar** | 2 | 3 | 5 | 9 | 10 |
| | 2 | 3 | 3 | 6 | 7 |
| | 2 | 3 | 4 | 6 | 6 |
| **Rata-Rata** | **2** | **3** | **4** | **7** | **7.7** |
| **Multi Scalar** | 1 | 4 | 5 | 8 | 12 |
| | 1 | 3 | 5 | 11 | 7 |
| | 2 | 4 | 3 | 6 | 7 |
| **Rata-Rata** | **1.3** | **3.7** | **4.3** | **8** | **8.7** |
| **Tercepat** | Scalar | Cross-Product Scalar | Cross-Product Scalar | Cross-Product Scalar | Cross-Product Scalar |
| **Terlama** | Cross-Product Scalar | Multi Scalar | Correlated | Correlated | Scalar |

Dari pengujian empat jenis query pada tiga tabel seperti pada Tabel 10, jenis *query cross-product scalar* memiliki rata-rata *elapsed time* paling cepat dengan jumlah data lebih dari 20 baris dibandingkan *elapsed time* dari tiga jenis query lainnya. Sedangkan *query scalar* memiliki *elapsed time* tercepat pada jumlah data 20 baris.

**Tabel 11. Perbandingan query pada 4 tabel**

| **Query** | **Jumlah Data (baris)** | | | | |
|---|---|---|---|---|---|
| | 20 | 40 | 60 | 80 | 100 |
| | **Elapsed Time (ms)** | | | | |
| **Scalar** | 2 | 3 | 5 | 6 | 12 |
| | 1 | 3 | 6 | 8 | 7 |

| | | | | | |
|---|---|---|---|---|---|
| | 1 | 4 | 5 | 6 | 7 |
| **Rata-Rata** | **1.3** | **3.3** | **5.3** | **6.7** | **8.7** |
| **Correlated** | 1 | 2 | 8 | 7 | 6 |
| | 1 | 2 | 6 | 9 | 5 |
| | 1 | 2 | 5 | 8 | 5 |
| **Rata-Rata** | **1** | **2** | **6.3** | **8** | **5.3** |
| **Cross-Product Scalar** | 1 | 2 | 3 | 3 | 7 |
| | 3 | 2 | 0 | 1 | 3 |
| | 1 | 2 | 3 | 2 | 3 |
| **Rata-Rata** | **1.7** | **2** | **2** | **2** | **4.3** |
| **Multi Scalar** | 2 | 2 | 2 | 5 | 8 |
| | 1 | 0 | 3 | 4 | 3 |
| | 1 | 2 | 3 | 2 | 3 |
| **Rata-Rata** | **1.3** | **1.3** | **2.7** | **3.7** | **4.7** |
| **Tercepat** | Correlated | Multi Scalar | Cross-Product Scalar | Cross-Product Scalar | Cross-Product Scalar |
| **Terlama** | Cross-Product Scalar | Scalar | Correlated | Correlated | Scalar |

Dari pengujian empat jenis query pada empat tabel seperti pada Tabel 11, jenis query *cross-product scalar* memiliki *elapsed time* tercepat pada jumlah data 60 baris hingga 100 baris. Pada pengujian dengan jumlah data 20 baris, *query correlated* memiliki *elapsed time* tercepat. Sedangkan pada pengujian dengan jumlah data 40 baris, *query multi scalar* memiliki *elapsed time* tercepat.

## 4. KESIMPULAN DAN SARAN

Berdasarkan pengujian yang telah dilakukan terhadap *elapsed time* dari eksekusi empat jenis query pada dua hingga empat tabel di Oracle, dapat dinyatakan bahwa jenis *query correlated* lebih optimal digunakan untuk jumlah relasi yang sedikit seperti relasi antara dua tabel. Sedangkan jenis *query cross-product scalar* dapat dieksekusi lebih optimal dibandingkan jenis query lainnya dengan jumlah data jumlah relasi yang banyak. Tidak disarankan untuk menggunakan *query correlated* dan *scalar* dalam mengeksekusi data dan relasi dengan jumlah besar.

## 5. DAFTAR RUJUKAN

[1] Suendri S., 2019. Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan Sistem Informasi Remunerasi Dosen Dengan Database Oracle (Studi Kasus:

UIN Sumatera Utara Medan). *Algoritma: Jurnal Ilmu Komputer Dan Informatika*, 2(2), pp.2.

[2] Praba, A. D. dan Safitri, M., 2020. Studi perbandingan performansi antara mysql dan postgresql. *Jurnal Khatulistiwa Informatika*, 8(2), pp-.

[3] Silalahi, M., 2018. Perbandingan performansi database mongodb dan mysql dalam aplikasi file multimedia berbasis web. *Computer Based Information System Journal*, 6(1). Pp 63.

[4] Noviyanti, P., dkk., 2018. Perbandingan Query Response Time pada Model Query View dan Cross Product. *E-JURNAL JUSITI: Jurnal Sistem Informasi dan Teknologi Informasi*, 7(2), pp.131–141.

[5] Purwoko, H., 2018. Pemanfaatan Basis Data Oracle Pada Sistem Informasi Work Order Pada PT XYZ Di Jakarta Timur. *CESS (Journal of Computer Engineering, System and Science)*, 3(2), pp.117–121.

[6] Samidi, dkk., 2022. Optimasi Database Menggunakan SQL Kueri Klausa IN dan EXIST pada Database Oracle 12c. Studi Kasus pada Aplikasi di Badan Nasional Pencarian dan Pertolongan (BASARNAS). *Jurnal Pendidikan Tambusai*. 6(1), pp. 2297–2306.

[7] Syamsiar E., 2004. *Oracle 9i: Optimasi Database*. 1st ed. Surabaya: Elex Media Komputindo.

[8] Rochman dan Juwari, J., Analisis Perbandingan Cartesian Product, Cross Join, Inner Join dan Outer Join dalam Si Akad. *Techno (Jurnal Fakultas Teknik, Universitas Muhammadiyah Purwokerto)*. 19(1) pp.37–44.

[9] Nurjaman, J., Perbandingan Optimasi Query Menggunakan Query Scalar, Correlated Dan Kombinasi, *Jurnal Bangkit Indonesia*. 5(1), pp. 20.

[10] Tavares , O. M. I., dkk., Analisis Model Restrukturisasi Kueri dalam Mengoptimasi Waktu Respons Eksekusi Data Pada Basis Data Relasional Mysql Php 7.2. 27.