Tuning Database Pada Sistem Penerimaan Mahasiswa Baru Menggunakan Optimasi Query dan Indexing

Tuning Database on New Student Admission System Using Query and Indexing Optimization

Amat Deska[1], Ipal Akbar[2], Samidi[3]
[1,2,3]Program Studi Magister Ilmu Komputer, Fakultas Teknologi Informasi, Universitas Budi Luhur
E-mail: [1]2111601932@student.budiluhur.ac.id, [2]2111601098@student.budiluhur.ac.id, [3]samidi@budiluhur.ac.id

Abstrak

Dalam pengoperasian database MaraiaDB diperlukan aplikasi berupa server localhost yang memiliki response waktu untuk menjalankan sebuah query agar dapat mendapatkan waktu yang efisiensi. Pada penelitian ini mengukur perfoma query dalam bentuk SELECT pada database MariaDB yang sudah di install pada komputer atau laptop dengan menggunakan aplikasi yang bernama Xammp dengan jumlah record data kurang lebih sebanyak 12.000 data, tapi pada penelitian ini hanya memakai sekitar 5000 data. Data tersebut nantinya akan dilakukan optimasi query dan indexing. Permasalahan yang terjadi adalah lamanya proses pengambilan data yang membutuhkan waktu sehingga membutuhkan suatu cara agar dapat mempercepat proses pengambilan data.Metode optimasi database yang difokuskan pada pengujian ini adalah dengan melakukan perbandingan dari berbagai efektivitas sub query serta penggunaan indexing pada tabel. Pada Query yang diuji adalah fungsi yang bernama LEFT JOIN, WHERE, ON, GROUP BY, ORDER BY dan DML (Data Manipulation Language), yaitu QUERY SELECT yang akan di lakukan pada aplikasi Xammp. Pada penelitian ini hasil yang di harapkan berupa pengambilan data yang akan menjadi lebih cepat atau efisien untuk meningkatkan kinerja pada database setelah melakukan Optimasi SQL Query dan Table Indexing.

Kata kunci: Tuning , Index, Optimasi Database, SQL Query

Abstract

In operating the MaraiaDB database, an application in the form of a localhost server is needed that has a response time to run a query in order to get time efficiency. In this study, we measure query performance in the form of SELECT on the MariaDB database that has been installed on a computer or laptop using an application called Xammp with a number of data records of approximately 12,000 rows, but in this study only used about 5000 rows. The data will later be optimized for querying and indexing. The problem that occurs is the length of the data retrieval process which takes time so it requires a way to speed up the data retrieval process.The database optimization method that is focused on this test is to compare the effectiveness of various sub-queries and use indexing on tables. The query being tested is a function called LEFT JOIN, WHERE, ON, GROUP BY, ORDER BY and DML (Data Manipulation Language), namely QUERY SELECT which will be performed on the Xammp appli-cation. In this research the results obtained in the form of data retrieval that becomes faster or more efficient to im-prove performance on the database after optimizing SQL Query and Table Indexing.Keyword : Tunning, Index, Optimation Database, Query SQL

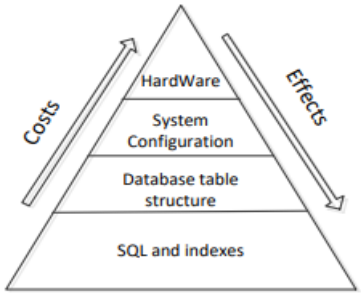
Keywords: Tuning, Index, Optimation Database, Query SQL

1. PENDAHULUAN

Pada sebuah organisasi sangat diperlukan adanya manajemen pengelolaan data, salah satu yang memiliki fungsi untuk mengelola data dengan baik yaitu DBMS (Database Management System). DBMS adalah program yang mengelola data di dalam database dan terstruktur dalam menyimpan sebuah data. Sistem dengan pengelolaan database yang baik merupakan tanda-tanda vital sebuah perusahaan mengandalkan sistem yang efektif dan efisien untuk menjalankan aktivitas sehari-hari [1]. Dengan banyaknya jumlah data yang dimiliki pada sistem database akan mengakibatkan lambatnya pada proses mengeksekusi perintah query SQL. Hal itu juga yang terjadi pada instansi pendidikan pada saat mengakses database.

Index adalah struktur data fisik yang terpisah yang memungkinkan query dapat mengakses satu atau lebih baris data dengan cepat [2][3]. Dengan Index maka, tidak perlu melakukan pencarian dari table baris awal hingga baris akhir yang akan membutuhkan waktu dalam proses pengambilannya[4]. Selain itu implementasi index juga diperlukan untuk mengurangi untuk menghilangkan error pada pengambilan data, mempercepat proses pengambilan data dan mengoptimalkan kinerja database [5].

Tuning adalah proses dalam menjalankan sebuah perintah dalam sistem dengan tujuan meningkatkan kinerja. Adapun manfaat dari tuning itu sendiri adalah untuk mempersingkat response time dan meningkatkan throughput. Tuning dibedakan mejadi 2 yaitu, Tuning for response time yang bertujuan untuk mempercepat pada saat proses penambilan data dan Tuning for throughput yang bertujuan untuk meningkatkan jumlah throughput process. Ada beberapa teknik untuk meningkatkan kinerja database agar mencapai level tertinggi. Proses tuning kinerja mencakup antara lain pengukuran respon waktu sebelum tuning, pada waktu tuning dilakukan, dan pengukuran respon waktu setelah tuning. [6]



Gambar 1 Hubungan pada SQL Optimasi [7]

Dapat dilihat dari gambar 1 bahwa metode umum pada optimasi database berdasarkan keefektivitasan dan tingkat biayanya. Optimasi perintah SQL dan optimasi indeks adalah metode yang paling murah tetapi justru metode yang paling efektif [7].

Elmasri dan Navathe pada bukunya [8] mengatakan Query memiliki banyak kemungkinan strategi eksekusi dan proses pindai yang cocok untuk diproses Query dikenal sebagai Optimasi Query. Penulisan SQL juga harus dilakukan secara sistematis agar mendapatkan keefektivitasan yang cepat dalam waktu responnya. Adapun menurut [9] Pengoptimal pada Query adalah inti dari kinerja RDBMS dan juga harus diperluas dengan pengetahuan tentang cara menjalankan fungsi yang ditentukan pengguna secara efisien, memanfaatkan struktur indeks baru, mengubah kueri menjadi cara baru dan bernavigasi diantara data menggunakan referensi. Menurut Zhang [10] Meskipun itu adalah tugas yang sangat kompleks tapi Mengoptimalkan sistem database memainkan peran yang sangat penting.

Pada beberapa penelitian sebelumnya terdapat referensi jurnal yang juga membahas mengenai tuning dan penggunaan indexing juga SQL optimasi Query. Seperti pada jurnal [11], jurnal ini membahas tentang membandingkan waktu respon database sebelum dan sesudah proses tuning database dengan menggunakan Optimasi Query dan Table Indexing. Adapun hasil dari penelitian ini adalah setelah dilakukan SQL Query, pada saat pengambilan data terjadi selisih yang sangat besar dimana membutuhkan waktu sekitar 743 detik pada sebelum tuning dan hanya

46 detik setelah dilakukan tuning dengan jumlah data sebanyak 1.039.852. Sedangkan pada penelitian lainnya menurut [6] bahwa Kinerja database dapat ditingkatkan dengan mengurangi tabel atau indeks fragmentasi. Ini didapati dengan memindahkan dan membuat ulang Tabel index, membuat ulang index dengan menggunakan perintah DBCC DBREINDEX atau dengan mengelompokkan indeks menggunakan perintah DBCC INDEXDEFRAG. Jurnal [12] membahas, memeriksa dan menjelaskan apa saja tools, teknik dan pada logika DBMS parameter yang berlaku untuk meningkatkan eksekusi query dalam lingkungan database. Tujuan optimasi database atau penyetelan SQL-Query, keduanya tekniknya sama, yaitu, mengurangi waktu respons dan konsumsi komputasi yang lebih sedikit & sumber daya jaringan. Hasil pada jurnal ini adalah penulis telah mempelajari dan menganalisis berbagai proses dan teknik untuk mengoptimalkan database dan menyetel SQL-Query. Penulis juga menilai bahwa ekspresi aljabar memiliki potensi tinggi dalam hal ini. Pada jurnal lainnya [13] membahas mengenai kinerja optimasi yang terkait dengan akses data dalam database yang berisi produk kedelai dan jagung untuk memastikan pencarian data secara cepat pada database. Dan hasilnya adalah teknik optimasi database telah diimplementasikan untuk mengatasi masalah yang dapat membatasi kinerja database ke tingkat kerentanan. Jurnal [14] memperkenalkan Neo (Neural Optimizer), yaitu pengoptimal kueri berbasis pembelajaran baru yang mengandalkan jaringan saraf dalam untuk menghasilkan rencana eksekusi kueri. Neo iteratif meningkatkan kinerjanya melalui kombinasi pembelajaran penguatan dan strategi pencarian. Hasilnya pengoptimal pembelajaran end-to-end pertama yang menghasilkan rencana eksekusi kueri yang sangat efisien menggunakan jaringan neural yang dalam. Serta pada jurnal [15] mempresentasikan Raven, sebuah sistem yang kami bangun untuk melakukan inferensi *Machine Learning* pada Database. Raven akan melakukan analisis statis dari pipeline Python dan kueri SQL, yang ditangkap dalam Unified Intermediate Representation (IR). Hasil penelitian ini menunjukkan pada data nyata yang mengalami peningkatan kinerja hingga 5,5 kali dari integrasi asli *Machine Learning* di SQL Server dan dari menerapkan optimasi silang baru, menghasilkan kinerja peningkatan kinerja hingga 24 kali. Dan pada jurnal [16] melakukan optimalisasi database dengan metode Index dan Partition Table. Dari hasil eksperimen dengan dataset sebesar 2.248.590 data didapatkan sebuah hasil yang sangat memuaskan dimana terdapat perbedaan waktu query yang signifikan sebelum dan sesudah menggunakan index dan partition table, dimana Database sebelum diberi Index dan Partition Table akan memakan waktu yang lama dibanding setelah menggunakan Indexdan Partition Table.

Dalam penelitian sebelumnya, menganalisis penurunan kinerja basis data, menggunakan metode pengindeksan dan optimisasi kueri dan menyatakan bahwa kinerja basis data dapat ditingkatkan dengan memodifikasi *query* SQL dan pengindeksan basis data, namun penelitiannya tidak mencakup pengujian eksperimental dan perbandingan hasil. Pada penelitian ini berhasil meningkatkan kinerja database dengan menerapkan optimasi query yang membuat proses query lebih cepat. Tujuan utama dari penelitian ini adalah untuk membandingkan waktu respon database sebelum dan sesudah proses tuning database juga membuktikan teori pada penelitian sebelumnya yang memperkuat metode penelitian ini.

2. METODE PENELITIAN

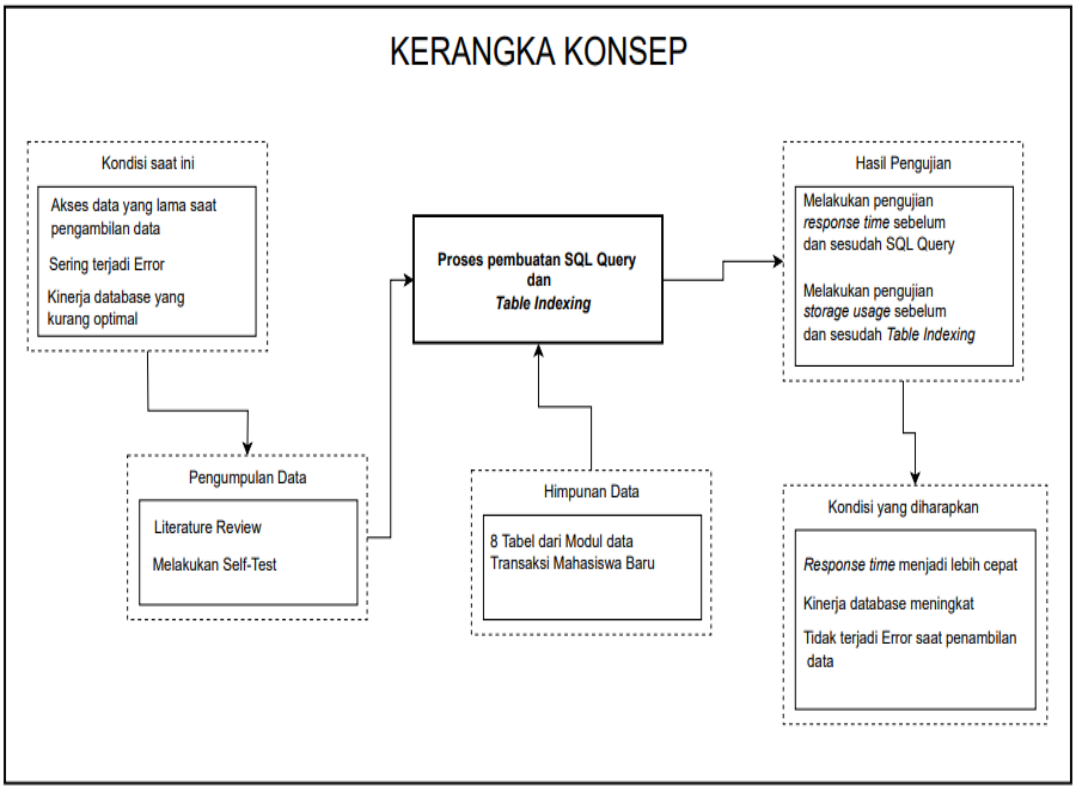
Database Manajemen Sistem adalah kumpulan dari program-program yang membolehkan user untuk menciptakan dan memelihara sebuah database. DBMS sudah menjadi peralatan standar untuk melindungi pengguna komputer dari bagian - bagian kecil dalam pengelolaan secondary storage. DBMS didesain untuk meningkatkan produktivitas dari aplikasi para programmer dan untuk memberikan kemudahan pengaksesan data oleh komputer. Pengaksesan data di dalam DBMS dapat dilakukan dengan berbagai macam cara. Dan tentunya dalam melakukan pengaksesan data ada hal-hal yang perlu diperhatikan seperti implementasi dari data itu sendiri serta waktu prosesnya. Ada banyak plan yang dapat diikuti oleh database manajemen sistem dalam memproses dan menghasilkan jawaban sebuah query. Semua plan pada akhirnya akan menghasilkan jawaban yang sama tetapi pasti mempunyai hal yang berbeda, seperti misalnya total waktu yang diperlukan untuk menjalankan sebuah query.

Optimisasi query mencoba memberikan suatu pemecahan untuk menangani masalah tersebut dengan cara menggabungkan sejumlah besar teknik-teknik dan strategi, yang meliputi transformasi logika dari query yang akan mengoptimisasi jalan akses dan penyimpanan data pada sistem file. Setelah ditransformasikan, sebuah query harus dipetakan ke dalam sebuah urutan operasi untuk menghasilkan data yang diminta.

Data sampel ini diambil dari data Pendaftaran pada tahun 2021/2022 yang terdiri dari 8 tabel. Nantinya data yang dihasilkan dalam modul ini akan menjadi sumber data utama untuk proses pembayaran formulir dan kuliah pada semester 1. Dalam percobaan ini jumlah sampel data yang diuji adalah sebanyak 5000 rows. Beberapa pengujian pada penelitian ini adalah dengan menjalankan perintah SQL yang dipilih pada tabel dan membandingkan hasil waktu respon yang didapatkan saat mengambil data dari perintah SQL tersebut. Ini dilakukan dengan dua kondisi berbeda yaitu: Sebeum proses optimasi dan Sesudah proses Optimasi.

Adapun langkah-langkah yang dilakukan pada pengujian ini yaitu: 1. Mengekstraksi data dari beberapa tabel yang dibutuhkan pada database Instansi Pendidikan; 2. Mentransformasi data; 3. Menload data kedalam database sampel MariaDB; 4. Menjalankan perintah SQL Commad dan mencatat hasil dari tes pertama; 5. Melakukan Optimasi SQL Query; 6. Proses Table Indexin, yaitu menentukan dan membuat indeks pada semua tabel; 7. Menjalankan perintah SQL Commad dan mencatat hasil kedua; Dan langkah terakhir adalah 8. Membandingkan antara hasil tes yang pertama dan kedua.

Sedangkan kerangka konsep yang digunakan pada penelitian ini adalah sebagai berikut:

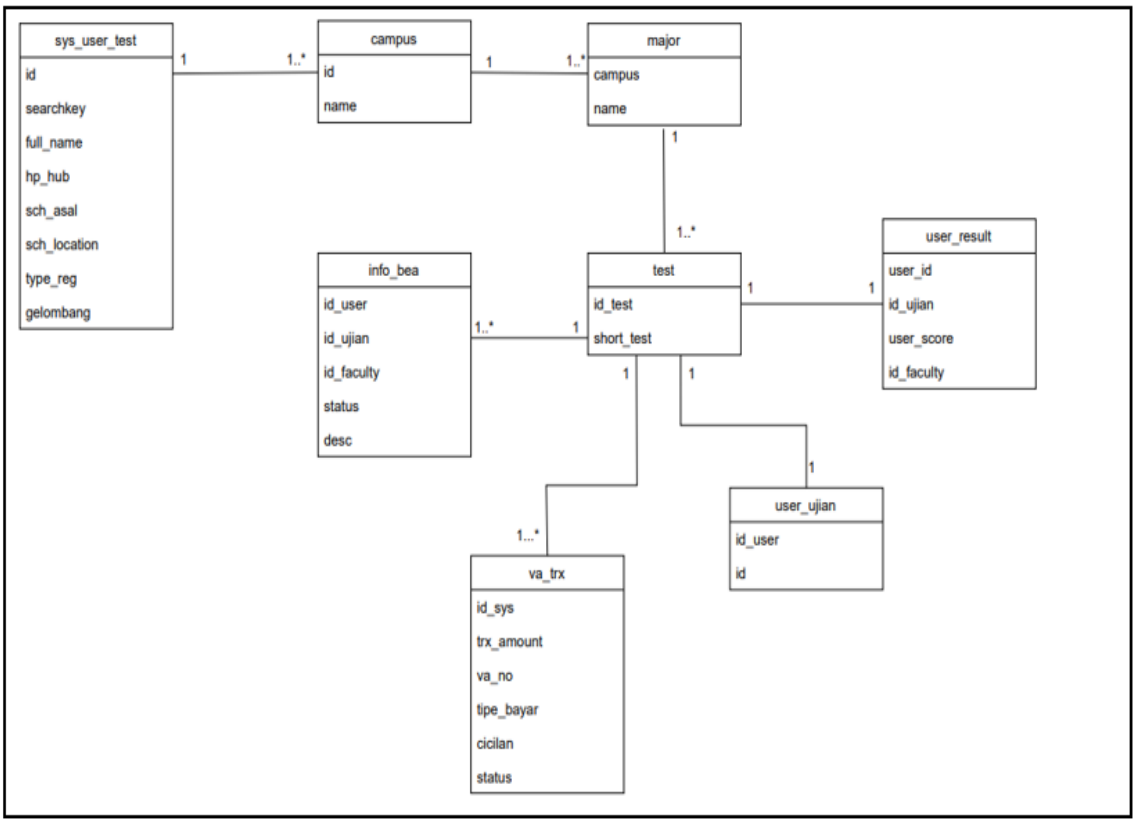


Gambar 2 Kerangka Konsep

Pada pengujian ini menggunakan perangkat dengan spesifikasi sebagai berikut : Sistem Operasi Windows 10 Pro 64 Bit, Processor Intel Core i5-3330, CPU; 3.00GHz Ram 8GB, Hardisk 500Gb, MariaDB 10.4.24 dan Xampp 7.4.29

3. HASIL DAN PEMBAHASAN

Pada objek peneltian ini terdapat 8 tabel yang saling berelasi pada modul data transaksi Pendaftaran Mahasiswa baru. Tabelnya adalah sys_user_test, campus, major, test, info_bea, va_trx, user_ujian dan user_result yang digambarkan dalam Logical Record Structurde (LRS) seperti yang ditunjukkan pada gambar di bawah ini.



Gambar 3 Logical Record Structurde (LRS)

Perbedaan database yang digunakan sebagai perbandingan pengujian antara data yang dilakukan query dan sebelum dilakukan query adalah sebagai berikut:

Pada Query dibawah ini bertujuan untuk memanggil data dalam kurun waktu hanya 1 bulan, dan akan mendapatkan waktu perbedaan pada saat melakukan eksekusi pada SQL tersebut.

Tabel 1 Query SQL sebelum dan sesudah Optimasi

BEFORE	**AFTER**
Query 1	
SELECT a.create_time, a.full_nme, a.*, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.*, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f	SELECT a.create_time, a.full_nme, a.sch_asal, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.user_score, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f

ON f.id=c.id_campus WHERE MONTH(a.create_time) = '3' GROUP BY full_nme ORDER BY a.create_time DESC;	ON f.id=c.id_campus where a.create_time BETWEEN '2021-03-01 00:00:00' and '2021-03-31 23:59:59' GROUP BY full_nme ORDER BY a.create_time ASC;

Pada Query kedua ini bertujuan untuk memanggil data dalam waktu 2 bulan, dan akan mendapatkan waktu perbedaan pada saat melakukan eksekusi pada SQL tersebut.

Tabel 2 Query SQL sebelum dan sesudah Optimasi

BEFORE	**AFTER**
Query 2	
SELECT a.create_time, a.full_nme, a.*, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.*, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f ON f.id=c.id_campus where MONTH(a.create_time) BETWEEN '3' AND '4' GROUP BY full_nme ORDER BY a.create_time ASC;	SELECT a.create_time, a.full_nme, a.sch_asal, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.user_score, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f ON f.id=c.id_campus where a.create_time BETWEEN '2021-03-01 00:00:00' and '2021-04-31 23:59:59' GROUP BY full_nme ORDER BY a.create_time ASC;

Pada Query ketiga ini bertujuan untuk memanggil data dalam waktu 4 bulan, dan akan mendapatkan waktu perbedaan pada saat melakukan eksekusi pada SQL tersebut.

Tabel 3 Query SQL sebelum dan sesudah Optimasi

BEFORE	**AFTER**
Query 3	
SELECT a.create_time, a.full_nme, a.*, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.*, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f ON f.id=c.id_campus WHERE MONTH(a.create_time) BETWEEN '3' AND '4' OR MONTH (a.create_time) BETWEEN '5' AND '6' GROUP BY full_nme ORDER BY a.create_time DESC;	SELECT a.create_time, a.full_nme, a.sch_asal, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.user_score, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f ON f.id=c.id_campus where a.create_time BETWEEN '2021-03-01 00:00:00' and '2021-06-31 23:59:59' GROUP BY full_nme ORDER BY a.create_time ASC;

Pada Query ke empat ini bertujuan untuk memanggil data dalam waktu 5 bulan, dan akan mendapatkan waktu perbedaan pada saat melakukan eksekusi pada SQL tersebut.

Tabel 4 Query SQL sebelum dan sesudah Optimasi

BEFORE	AFTER
Query 3	
SELECT a.create_time, a.full_nme, a.*, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.*, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f ON f.id=c.id_campus WHERE MONTH(a.create_time) BETWEEN '3' AND '4' OR MONTH (a.create_time) BETWEEN '5' and '6' OR MONTH (a.create_time) = '7' GROUP BY full_nme ORDER BY a.create_time DESC;	SELECT a.create_time, a.full_nme, a.sch_asal, e.name as prodi, a.hp_num, a.email, f.name as kampus, d.user_score, IF(d.hasil = 1, "LULUS", "TIDAK LULUS") as keterangan FROM sys_user_test AS a JOIN user_ujian AS b ON a.id=b.id_user JOIN pilihan AS c ON b.id_user=c.id_sys_user LEFT JOIN user_result AS d ON b.id_user=d.user_id LEFT JOIN major AS e ON e.id=c.id_major LEFT JOIN campus AS f ON f.id=c.id_campus where a.create_time BETWEEN '2021-03-01 00:00:00' and '2021-07-31 23:59:59' GROUP BY full_nme ORDER BY a.create_time ASC;

Setelah melakukan eksekusi pada SQL Query, tahap berikutnya melakukan indexing, pada tahap indexing bertujuan untuk memangkas waktu lebih cepat.

Pada penelitian ini, perintah SQL yang dibuat digunakan untuk memanggil data transaksi pembayaran formulir dan kuliah pada semester 1.

Pada penelitian ini juga melakukan sebuah proses pada Query Optimization dengan melakukan perubahan pada perintah SQL. Perubahan pertama menghapus semua tanda "*" yang terdapat pada "SELECT" karena dapat memperlambat proses Query. Tanda pada "*" pada SQL akan di ganti dengan dengan nama kolom yang diperlukan. Contoh nya pada SQL diatas "a.*" di ganti dengan "a.sch_location" karena yang benar dalam query hanya nama sekolah, kemudian pada SQL "d.*" di ubah menjadi "d.user_score" seharus nya yang diperlukan hanya hasil score.

Perubahan kedua pada bagian "WHERE CLAUSE", untuk membatasi rentang waktu untuk melakukan pembayaran sebelum optimasi, digunakan dengan kondisi "WHERE MONTH(a.create_time) = 3 ", penggunakan kondisi ini membuat proses pemanggilan data lebih lambat dengan kondisi ini menjadi "WHERE a.create_time BETWEEN '2021-03-1 00:00:00' AND 2021-03-31 23:59:59".

Pada penelitian ini dibuat index pada beberapa tabel agar proses pengambilan datanya dapat memangkas waktu menjadi lebih cepat. Index yang dibuat seperti pada Tabel 2 dibawah ini. Berikut perintah SQL yang dijalankan untuk membuat Index dalam penelitian ini yaitu :

```
CREATE INDEX idx_id ON sys_user_test (id);
CREATE INDEX idx_user_id ON user_result (user_id);
CREATE INDEX idx_id_sys ON va_trx (id_sys);
```

Tabel 5 Perintah SQL dalam pembuatan Tabel Index

Tables	**Coloumns**
'sys_user_test'	'id', 'searchkey', 'full_nme', 'hp_hub', 'sch_asal', 'sch_location', 'type_reg', 'gelombang', 'create_time'.
'campus'	'id', 'name'.
'major'	'campus', 'name'.
'info_bea'	'id_ujian', 'status', 'id_faculty', 'id_user', 'desc'.
'user_result'	'user_id', 'id_ujian', 'user_score', 'id_faculty'.
'user_ujian'	'id_user', 'id'.
'test'	'id_test', 'short_desc'.
'va_trx'	'id_sys', 'trx_amount', 'va_no', 'tipe_bayar', 'cicilan', 'status'.

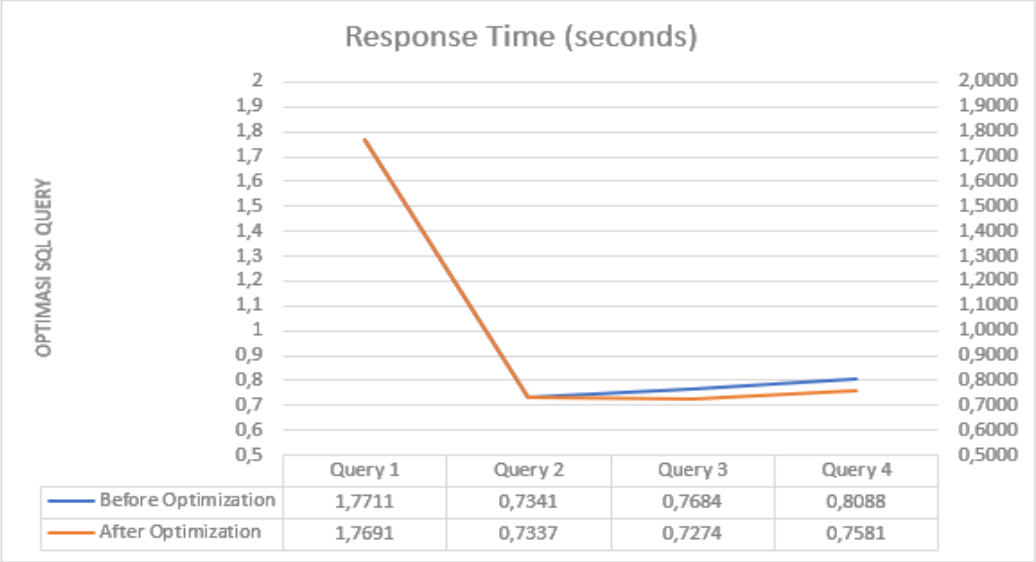
Semua Primary Key dan Foreign Key sudah terindeks. Contoh nya adalah pada kolom 'id' pada table 'sys_user_test', kemudian pada kolom terbaca pada kondisi 'WHERE CLAUSE' juga terindex sehingga proses melakukan QUERY menjadi lebih cepat, misal nya pada kolom 'id_sys' pada tabel 'va_trx'.

Pada database terdapat 1 tabel transaksi dengan data sebagai berikut :
Va_trx : 21,417 rows memakan data 5,5 MB (Mega Byte)

Data yang dipanggil pada saat pengujian memiliki jumlah yang berbeda-beda , sehingga dapat dilihat waktu respons pada saat memanggil data sebanyak 579 rows. Diperlukan waktu selama 1,7711 seconds sebelum optimasi dan setelah optimasi menjadi 1,7691 seconds. Selanjutnya pada query yang kedua jumlah data yang dipanggil sebanyak 1194 rows, sebelum optimasi membutuhkan waktu 0,7341 seconds sebelum optimasi dan 0,7337 seconds setelah dilakukan optimasi. Query ketiga memanggil data sebanyak 3677 rows pada saat sebelu dioptimasi membutuhkan 0,7684 seconds dan setelah dioptimasi menjadi 0,7274 seconds. Dan untuk Query yang keempat jumlah data yang dipanggil sebanyak 5565 rows, waktu yang diperlukan untuk memanggilnya sebelum dioptimasi yaitu 0,8088 seconds dan 0,7581 seconds setelah dioptimasi. Hasil dai pengujian tadi dapat dilihat pada tabel dan gambar dibawah ini.

Tabel 5 Hasil Pengujian pada SQL Query

SQL Query	**DATA ROWS (records)**	**BEFORE (seconds)**	**AFTER (seconds)**	**DIFFERENCE**
Query 1	579	1,7711	1,7691	0,11%
Query 2	1194	0,7341	0,7337	0,05%
Query 3	3677	0,7684	0,7247	5,68%
Query 4	5565	0,8088	0,7581	6,26%



Gambar 4 Grafik perbedaan Respons Time sebelum dan sesudah Optimasi

4. KESIMPULAN DAN SARAN

Pada pengujian ini menunjukkan hasil perbedaan waktu respon pada saat menjalankan database ketika menggunakan perintah SQL sebelum dan sesudah optimasi dilakukan. Query pada saat sesudah dioptimasi lebih cepat dibandingkan Query sebelum dioptimasi. Hal ini membuktikan bahwa dengan melakukan Optimasi Query dan Table Indexing dapat memperkecil waktu respon dan meningkatkan kinerja pada database seperti pada penelitian sebelumnya [1] [6] [11] [12] [13] [15] [16] .

Harapannya penelitian ini tidak hanya diimplementasikan pada data transaksi pembayaran formulir dan kuliah di semester 1, tetapi juga pada semua modul yang ada pada Universitas agar diterapkan Tuning Database. Penelitian ini bisa dijadikan contoh untuk modul lain yang juga memiliki jumlah data yang besar seperti Data Mahasiswa dan Matakuliah.

Untuk kedepannya, penulis mengusulkan penelitian databse yang lain agar mengukur secara rinci beban dan biaya pada saat ingin melakukan Table Indexing karena pengindeksan ini membutuhkan sekitar 5%-15% media penyimpanan tambahan [17]. Pada akhirnya hal yang harus diperhatikan dalam penelitian ini adalah agar dilakukan dengan efisien karena berkaitan dengan kemungkinan mengekstraksi subset dari dataset besar secara efisien. Kompleksitas dan heterogenitas data, yang mencakup data terstruktur, semi-terstruktur, dan tidak terstruktur dapat membuat pengoptimalan kueri menjadi sulit [18].

DAFTAR PUSTAKA

[1] A. Abbas and K. Ahmad, “Query Performance in Database Operation,” PS-FTSM-2020-045, 2020.
[2] Kristina, Wasino, and Tony, “Perbandingan Optimasi Query dengan View dan Indexed View,” J. Ilmu Komput. dan Sist. Informas, pp. 113–117, 2013.
[3] D. Petkovic, “Microsoft SQL Server 2008 , A Beginner ’ s Guide,” no. August, 2008.
[4] R. Pamungkas, “Optimalisasi Query Dalam Basis Data My Sql Menggunakan Index,” Res. Comput. Inf. Syst. Technol. Manag., vol. 1, no. 1, p. 27, 2018, doi: 10.25273/research.v1i1.2453.
[5] S. Mukherjee, “Indexes in Microsoft SQL Server,” SSRN Electron. J., no. March, 2019, doi: 10.2139/ssrn.3415957.

[6] C. Cioloca and M. Georgescu, "Increasing Database Performance using Indexes," Database Syst. J., vol. 2, no. 2, pp. 13–22, 2011, [Online]. Available: https://ideas.repec.org/a/aes/dbjour/v2y2011i2p13-22.html.
[7] B. J. Liu, L. J. Wu, W. He, X. Y. Han, and L. L. Tang, "Research on Performance Optimization Technology of Complex Equipment Software Database," IOP Conf. Ser. Mater. Sci. Eng., vol. 1043, no. 2, pp. 0–7, 2021, doi: 10.1088/1757-899X/1043/2/022022.
[8] Ramez Elmasri and Shamkant B. Navathe, Fundamental Of Database Systems Seventh Edition, 7th ed. Pearson, 2016.
[9] T. Connolly and C. Begg, Database Systems A Practical Approach to Design, Implementation, and Management SIXth edition. 2014.
[10] J. Zhang, "Research on Database Application Performance Optimization Method," 6th Int. Conf. Mach. Mater. Environ. Biotechnol. Comput., no. Mmebc, pp. 2236–2239, 2016, doi: 10.2991/mmebc-16.2016.448.
[11] Samidi, D. Iskandar, M. Fachruroji, W. A. Septyo, and A. K. A, "Database Tuning in Hospital Applications Using Table Indexing and Query Optimization," J. Pendidik. Tambusai, vol. 6, pp. 1960–1967, 2022.
[12] M. Alam, N. Kumar, and S. B. Singh, "A Study of Various Parameters and Techniques Applicable in Database and," vol. XV, no. Viii, pp. 71–81, 2021.
[13] D. A. Domokos, D. Pamfil, and M. Grebenisan, "Database optimization techniques applied to a database contain soy and corn based products to ensure a quick search in this database.," Bull. Univ. Agric. Sci. Vet. Med. Cluj-Napoca. Hortic., vol. 69, no. 2, pp. 422–426, 2012.
[14] R. Marcus *et al.*, "Neo: A Learned query optimizer," *Proc. VLDB Endow.*, vol. 12, no. 11, pp. 1705–1718, 2018, doi: 10.14778/3342263.3342644.
[15] K. Karanasos *et al.*, "Extending Relational Query Processing with ML Inference," 2019, [Online]. Available: http://arxiv.org/abs/1911.00231.
[16] Samidi, Fadly, Y. Firmansyah, S. Y. Ronal, and A. B. Lesmana, "Optimasi Database dengan Metode Index dan Partisi Tabel Database Postgresql pada Aplikasi E-Commerce . Studi pada Aplikasi Tokopintar," vol. 6, no. 1, pp. 2094–2102, 2022.
[17] J. Yu and M. Sarwat, "Two birds, one stone: A fast, yet lightweight, indexing scheme for modern database systems," Proc. VLDB Endow., vol. 10, no. 4, pp. 385–396, 2016, doi: 10.14778/3025111.3025120.
[18] S. Anuja and C. Malathy, "Big Data Query Optimization -Literature Survey," pp. 1–9, 2021, doi: https://doi.org/10.21203/rs.3.rs-655386/v1.