Jurnal SimanteC
Vol. 11, No. 2 Juni 2023
P-ISSN : 2088-2130
E-ISSN : 2502-4884

PERBANDINGAN KINERJA *QUERY* SQL *JOIN TABLES* DENGAN MENGGUNAKAN *INDEX*

PERFORMANCE COMPARISON OF QUERY SQL JOIN TABLES USING INDEX

Kurniawan Eka Permana[1)], Moch Kautsar Sophan[2)], Arif Muntasa[3)], Abdullah Basuki Rahmat[4)]

[1,2,3,4]Prodi Teknik Informatika, Fakultas Teknik, Universitas Trunojoyo
Jl. Raya Telang, PO BOX 2 Kamal, Bangkalan
E-mail: [1*]kurniawan@trunojoyo.ac.id,
[2]kautsar@trunojoyo.ac.id,[3]arifmuntasa@trunojoyo.ac.id,[4]abdullahbasuki@trunojoyo.ac.id
**Corresponding author email.*

ABSTRAK

Desain database tanpa index berdampak pada query SQL tidak optimal sehingga waktu eksekusi aplikasi yang tinggi. Optimalisasi Query SQL memainkan peran penting dalam meningkatkan kinerja sistem basis data. Salah satu teknik utama untuk pengoptimalan query sql adalah penggunaan indeks. Indeks adalah struktur data yang memungkinkan pengambilan data yang efisien dengan menyediakan akses cepat ke subset data tertentu. Penelitian ini bertujuan untuk membandingkan dampak penggunaan indeks dengan tidak menggunakan indeks pada optimasi query sql khususnya pada tabel gabungan. Metode perbandingan menggunakan pengujian kecepatan kueri. Ada tiga skenario pengujian yang dilakukan. Skenario pertama adalah pengujian dengan satu gabungan pada dua tabel. Skenario kedua menguji tiga join pada empat tabel dan skenario ketiga menguji tiga join pada empat tabel dengan kondisi. Setiap skenario dilakukan pada tabel yang diindeks dan tabel yang tidak diindeks. Hasil pengujian menunjukkan bahwa query dengan indeks mengalami peningkatan kecepatan dibandingkan dengan query tanpa indeks. Pada tiga skenario percobaan diperoleh peningkatan kecepatan waktu query masing-masing sebesar 39,474%, 5,5% dan 43,68%. Menggunakan indeks dapat sangat meningkatkan kinerja query sql dengan mengurangi jumlah data yang diakses dan memungkinkan pengoptimalan query sql yang lebih efisien. Tanpa indeks, query sql mungkin memerlukan pemindaian tabel lengkap, yang menyebabkan waktu eksekusi lebih lambat, terutama untuk tabel yang lebih besar.

Kata kunci: *Index, Join Tables, Query Performance, SQL Query.*

ABSTRACT

Database design without an index has an impact on SQL queries that are not optimal so that the application execution time is high. Query optimization plays a crucial role in improving the performance of database systems. One of the key techniques for query optimization is the use of indexes. Indexes are data structures that enable efficient data retrieval by providing quick access to specific data subsets. The purpose of this research compares the impact of using indexes versus not using indexes on query optimization especially in join tables. The comparison method uses query speed testing. There are three test scenarios performed. The first scenario is testing with one join on two tables. The second scenario is testing three joins on four tables and the third is testing three joins on four tables with conditions. Each scenario is performed on indexed tables and non-indexed tables. The results of the test show that queries with an index experience an increase in speed compared to queries without an index. In three experimental scenarios, an increase in query time speed was obtained, respectively 39.474%, 5.5% and 43.68%. Using an index can greatly improve the performance of queries by reducing the amount of data accessed and allowing for more efficient query optimization. Without an index, queries may require full table scans, leading to slower execution times, especially for larger tables.

Keywords: *Index, Join Tables, Query Performance, SQL Query.*

PENDAHULUAN

Dalam sebuah aplikasi Sistem Informasi dengan penyimpanan database, kegiatan yang utama adalah kegiatan menyimpan, mengelola, dan mengakses data pada database. Salah satu engine database favorit yang sering digunakan adalah mySQL. Database MySQL adalah komponen penting yang berperan melakukan aktifitas *Create, Read, Update, Delete* data[1], [2]. Untuk memastikan *user experience* yang baik, maka harus dipastikan bahwa database dapat melakukan respons cepat, dan efisiensi operasional. Untuk itu harus dilakukan proses tune up kinerja database mySQL. Mengoptimalkan kinerja database MySQL merupakan langkah penting untuk memastikan bahwa sistem database berjalan secara efisien dan dapat menangani banyak beban kerja sistem[1]–[5].

Ada beberapa alasan yang menjelaskan mengapa mengoptimalkan kinerja database MySQL sangat penting yaitu kecepatan aplikasi, efisiensi sumber daya server, skalabilitas aplikasi, dan kehandalan aplikasi.

Kecepatan aplikasi atau *Application Responsiveness* akan mempengarahui kepuasan pengguna terhadap aplikasi[6]. Kualitas database MySQL yang buruk dapat memengaruhi daya tanggap aplikasi, sehingga aplikasi berjalan lambat. Pengguna mungkin tidak mendapatkan hasil yang diharapkan jika query sql database membutuhkan waktu lama untuk dieksekusi. Waktu respons yang lambat dapat membuat aplikasi tidak efektif dan membuat pengguna bosan.

Efisiensi Sumber Daya aplikasi akan mempengaruhi kinerja aplikasi[7]. Optimalisasi kinerja database MySQL membantu mengoptimalkan penggunaan sumber daya server seperti CPU, memori, dan disk. Mengoptimalkan penggunaan sumber daya server dapat mengurangi biaya perangkat keras yang diperlukan untuk menjalankan aplikasi dengan mengurangi beban kerja yang tidak perlu dan mempercepat eksekusi query.

Aplikasi harus memiliki Skalabilitas yang fleksibel, artinya kapasitas aplikasi harus bisa ditingkatkan dengan mudah[8]. Mengoptimalkan kinerja database MySQL sangat penting untuk mempertahankan skalabilitas sistem dalam situasi di mana data berkembang dengan cepat atau lonjakan traffic data secara tiba-tiba. Database dapat mengatasi peningkatan beban kerja dengan meningkatkan kinerja jika kinerja mySQL optimal.

Aplikasi harus Reliable dan Available. Keandalan dan ketersediaan sistem dapat ditingkatkan melalui optimalisasi kinerja database MySQL. Jika *query* database membutuhkan waktu lama untuk dijalankan, maka proses update data dan operasi penting lainnya mungkin tertunda. Database lebih efisien dapat memberikan akses data yang cepat dan akurat, meminimalkan kesalahan sistem dan ketidaktersediaan sistem, pada akhirnya akan mengurangi waktu tunggu sistem.

Aplikasi harus dapat memberikan *user experience* yang memuaskan[9]. Pengalaman pengguna secara langsung dipengaruhi oleh pengoptimalan kinerja database MySQL. Pengguna akan puas dengan aplikasi dan akan meningkatkan kepercayaan mereka terhadap sistem dengan memastikan database memiliki respon yang cepat. Dengan menggunakan database dengan benar, pengembang aplikasi / *software developer* dapat memberikan fitur dan fitur yang lebih canggih kepada pengguna, dan akan meningkatkan value aplikasi.

Untuk memastikan aplikasi yang responsif, efisien, dan andal, maka mengoptimalkan kinerja database MySQL adalah hal yang sangat penting. Organisasi dapat meningkatkan produktivitas, meningkatkan kepuasan pengguna, dan mengurangi risiko yang berkaitaan dengan kinerja aplikasi yang buruk dengan mengoptimalkan kinerja database[10]–[12].

Salah satu cara penting untuk mengoptimalkan kinerja database adalah dengan menggunakan indeks. Indeks

database MySQL akan mempercepat akses data dengan mengurangi waktu yang dibutuhkan untuk operasi seperti pencarian, pengurutan, dan penggabungan[13], [14].

Database MySQL menggunakan struktur data yang disebut B-Tree (Binary Tree). B-Tree adalah struktur data yang dimaksudkan untuk mempermudah penyimpanan dan akses data. MySQL akan membuat dan memelihara struktur B-Tree khusunya yang berkaitan dengan indeks pada kolom atau kolom tertentu dalam sebuah tabel[10], [12]. Metode indeks dapat juga digunakan untuk menghindari pencarian berurutan (full scan) dari kedua tabel yang sedang di lakukan query join.

Namun, ada beberapa hal yang harus diperhatikan saat menggunakan indeks di mySQL, yaitu: 1. Storage overhead: Indeks membutuhkan lebih banyak ruang penyimpanan database. Setiap indeks yang digunakan akan meningkatkan ukuran database dan memengaruhi jumlah waktu yang diperlukan untuk operasi seperti insert, update, dan delete data. 2. Maintenance Cost: Saat melakukan insert, update, dan delete data, MySQL perlu mempertahankan indeksnya. Maintenance Cost dapat meningkat jika ada banyak indeks, yang berdampak pada kinerja secara keseluruhan, terutama pada saat proses Insert, delete, dan update. 3. Memilih jenis indeks yang tepat: Memilih kolom yang tepat untuk diindeks serta jenis indeks yang tepat sangatlah penting. Memilih indeks yang salah dapat menyebabkan penggunaan sumber daya yang berlebihan atau bahkan penurunan kinerja database mySQL[15].

Untuk memastikan indeks tetap efektif, diperlukan pemantauan, analisis, dan pemeliharaan rutin. Untuk merancang dan mengoptimalkan indeks dengan baik, kita harus mengetahui bagaimana data diakses dan apa yang dilakukan terhadap database. Dalam penelitian ini kami mencoba meningkatkan performa database MySQL yang akan berdampak pada respons aplikasi secara keseluruhan dengan menggunakan indeks dengan bertahap.

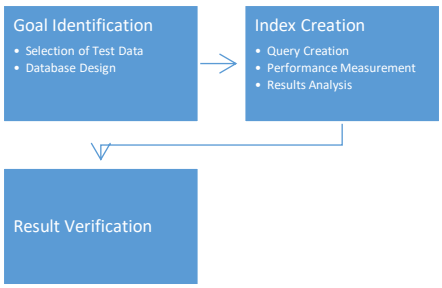
Tujuan dari penelitian ini adalah untuk mengetahui bagaimana penggunaan indeks pada database MySQL dapat mempengaruhi kinerja database terutama pada data transaksi dengan jumlah record banyak.

Manfaat penelitian ini untuk meningkatkan pemahaman akan pentingnya penggunaan indeks pada database MySQL dan memberikan panduan praktis untuk desain dan optimalisasi indeks guna meningkatkan kinerja basis data.

Dalam penelitian ini kami menggunakan database mysql dataset yang kami dapatkan dari https://github.com/datacharmer/test_db. Database ini adalah database pegawai (salaries), department, histori department pegawai (employee), dan gaji pegawai. Total baris data gaji pegawai adalah 2.844.047.

METODE

Salah satu bentuk pendekatan penelitian adalah metode eksperimen[16]. Metode ini digunakan dalam penelitian ini. Peneliti menggunakan pendekatan eksperimental untuk memodifikasi dan menyesuaikan faktor yang mempengaruhi hasil eksperimen dan mempelajari hasilnya. Pendekatan eksperimental yang peneliti lakukan dapat dilihat pada Gambar 1.



Gambar 1. Tahap Eksperimen

Metodologi pengujian indeks di MySQL yang kami lakukan melibatkan langkah-langkah berikut:

1. Identifikasi Tujuan Pengujian.
2. Pemilihan data uji.
3. Penyiapan Skema Database.

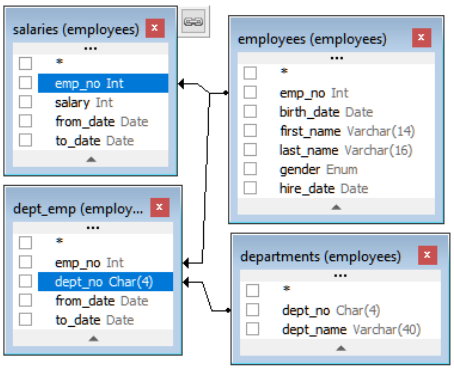
4. Pembuatan Indeks.
5. Pengujian *Query*.
6. Pengukuran kinerja.
7. Analisis Hasil.
8. Iterasi dan Verifikasi.

Tujuan pengujian indeks yang ingin dicapai peneliti adalah meneliti dengan menggunakan teknik pendekatan eksperimen dengan menguji dua kelompok query SQL yaitu satu kelompok menggunakan indeks dan satu kelompok tanpa menggunakan indeks.

Dalam pemilihan data uji peneliti menentukan data yang akan dilakukan pengujian. Data yang digunakan dalam percobaan ini adalah dataset sintetik atau dataset nyata yang diambil dari lingkungan produksi, yaitu di alamat url https://github.com/datacharmer/test_db. Eksperimen akan dilakukan pada platform database yang umum digunakan yaitu MySQL versi 8. Editor pengujian query dilakukan pada software Heidi SQL. Waktu eksekusi query akan diukur dan dibandingkan antara dua kelompok query yang menggunakan indeks dan tanpa menggunakan indeks.

Dalam tahap penyiapan skema basis data, peneliti menyiapkan struktur tabel sesuai dengan data yang akan diuji. Diagram struktur basis data yang digunakan dapat dilihat pada gambar 2. Skema database disesuaikan dengan data uji.

Database yang kami gunakan terdiri dari empat tabel yaitu tabel departement, tabel dept_emp, tabel employees dan tabel salaries. Tabel Departement dan employees adalah tabel master, sementara tabel dept_emp dan salaries adalah tabel transaksi. Tabel 1 menunjukkan jumlah data yang tersimpan pada setiap tabel.



Gambar 2. Skema Database

Tabel 1. Nama Tabel dan Jumlah Data

Table name	Records
employees	300024
departments	9
dept_emp	331603
salaries	2844047

Dalam tahap pembuatan indeks peneliti mendefinisikan indeks kolom setiap table sesuai dengan kolom yang dilibatkan dalam *sql query join* dan *sql query* pencarian yang digunakan. Pembuatan indeks menggunakan perangkat lunak Heidi SQL yang dilakukan dengan cara mengatur kolom sebagai indeks.

Dalam tahap pengujian *query* peneliti merancang sekumpulan *query* yang menggunakan perintah *SQL Join*. *SQL Join* adalah salah satu metode untuk melakukan pencarian data di database dengan menggabungkan kolom dari 2 buah tabel atau lebih. *Query join* ini mensimulasikan kasus penggunaan kolom yang diindeks dan *query* pada kolom yang tidak diindeks.

Ada tiga *query* yang digunakan dalam percobaan ini, yaitu *single join*, *triple join*, dan *triple join* dengan pencarian.

a. *Single Join*

Gambar 3 menunjukkan perintah *sql query* dengan menggabungkan dua tabel dengan menggunakan *SQL Join*. Tabel yang dimaksud adalah tabel Salaries dan Employee. Kunci penghubung antara dua tabel adalah emp_no. Pada *query* ini akan menampilkan kolom dari tabel salaries dan kolom dari tabel employee. *SQL*

query menggunakan join pada kolom emp_no dari tabel salaries dan dari tabel employee. *SQL Query* ini akan menunjukkan kecepatan hasil running *query* saat menggunakan indeks pada kolom emp_no dan saat tidak menggunakan indeks.

```
Select
    employees.salaries.emp_no,

employees.salaries.from_date,
    employees.salaries.salary,
    employees.salaries.to_date,
    employees.birth_date,
    employees.first_name,
    employees.last_name,
    employees.gender,
    employees.hire_date
From
    salaries Left Join
    employees                  On
employees.salaries.emp_no      =
employees.emp_no
```

Gambar 3. *SQL Query* dengan *single Join*

b. *Triple Join*

Sementara itu, gambar 4 menunjukkan *query* yang digunakan untuk mengukur kinerja penggunaan tiga *Join*. Empat tabel terlibat dalam operasi *Join* ini, yaitu tabel salaries, employee, dept_emp dan departement.

```
select
    employees.salaries.emp_no,

employees.salaries.from_date,
    employees.salaries.salary,
    employees.salaries.to_date,
    employees.birth_date,
    employees.first_name,
    employees.last_name,
    employees.gender,
    employees.hire_date,
    dept_emp.dept_no,
    dept_emp.from_date         As
from_date1,
    dept_emp.to_date           As
to_date1,
    departments.dept_name
From
    salaries Left Join
    employees                  On
employees.salaries.emp_no      =
employees.emp_no Left Join
    dept_emp On dept_emp.emp_no
= employees.emp_no Left Join
    departments                On
dept_emp.dept_no               =
departments.dept_no
```

Gambar 4. SQL Query dengan triple Join

Pada 4 tabel tersebut di gabungkan berdasarkan kolom emp_no dan dept_no yang ada di tabel employee, salaries, dept_emp, dan departments. Setiap kolom yang digunakan untuk join akan dibandingkan dengan menggunakan indeks dan tanpa menggunakan indeks.

c. *Triple Join* dengan kondisi pencarian.

Pada gambar 5, kita menggunakan *Query* dengan tiga *Join* dalam tabel salaries, employee, dept_emp, dan departement. Pada *Query* ini terdapat kondisi tambahan untuk menyeleksi data yaitu kondisi pencarian dengan nama departemen “produksi”.

```
Select
    employees.salaries.emp_no,

employees.salaries.from_date,
    employees.salaries.salary,
    employees.salaries.to_date,
    employees.birth_date,
    employees.first_name,
    employees.last_name,
    employees.gender,
    employees.hire_date,
    dept_emp.dept_no,
    dept_emp.from_date         As
from_date1,
    dept_emp.to_date           As
to_date1,
    departments.dept_name
From
    salaries Left Join
    employees                  On
employees.salaries.emp_no      =
employees.emp_no Left Join
    dept_emp On dept_emp.emp_no
= employees.emp_no Left Join
departments On dept_emp.dept_no
= departments.dept_no
Where departments.dept_name    =
'Finance'                      or
departments.dept_name          =
'Production'
```

Gambar 5. *SQL Query* dengan *triple Join* dan kondisi pencarian

Perintah *SQL query* pada gambar 5 merupakan perluasan dari perintah *query* pada gambar 4. Perluasan yang dilakukan adalah dengan menambahkan kondisi pencarian dengan kolom production. Pada kolom ini juga akan dibandingan kecepatan *query* saat menggunakan indeks dan tanpa menggunakan indeks.

Pengukuran kinerja dilakukan oleh peneliti dengan memanfaatkan informasi *running time* yang ada di Heidy sql saat

perintah *sql* di jalankan. *Query* pengujian dijalankan pada database MySQL dan waktu eksekusi dicatat untuk masing-masing query.

Analisis hasil waktu eksekusi *query* dilakukan pada database sebelum dan sesudah mengimplementasikan indeks. Perbandingan kinerja *query* dengan indeks dan tanpa indeks akan dianalisis.

Dalam eksperiman ini peneliti melakun iterasi dan verifikasi sebanyak 5 kali. Lima kali langkah pengujian dilakukan untuk tiga skenario penggunaan setiap query yang berbeda.

Metode yang dirancang oleh peneliti dilakukan secara konsisten dan mengulang pengujian secara objektif untuk mendapatkan pemahaman yang akurat tentang pengaruh indeks terhadap kinerja MySQL.

HASIL DAN PEMBAHASAN

Setelah dilakukan percobaan, hasilnya seperti terlihat pada tabel 2. Dari hasil tersebut terlihat perbandingan kinerja antara query join dengan index dan query join tanpa menggunakan index. Dalam skenario pertama, kami melakukan query select tabel dengan satu JOIN. Hasil yang diperoleh pada rata-rata kinerja hasil eksekusi query dengan indeks (T1A) sebesar 5,66 detik. Sedangkan percobaan pertama dengan query execution time index (T1B) didapatkan hasil sebesar 3,42575 detik. Dengan indeks, waktu eksekusi query JOIN dua tabel berkurang dari 5,66 detik menjadi hanya 3,43575 detik. Hal ini menunjukkan bahwa T1B jauh lebih cepat dieksekusi dengan peningkatan kecepatan sebesar 39,474% dibandingkan dengan T1A.

Dalam skenario kedua, query JOIN tabel dilakukan dengan tiga JOIN. Rata-rata hasil eksekusi query adalah 12,254 detik untuk eksperimen tanpa indeks (T2A) dan 11,57425 detik untuk eksperimen dengan indeks (T2B). Waktu query T2A yang ditingkatkan adalah 5,5% lebih cepat daripada waktu kueri T2B.

Sedangkan pada skenario ketiga, query dilakukan dengan menggabungkan tiga tabel yang diikuti dengan kondisi. Hasil yang diperoleh pada saat percobaan tanpa indeks (T3A) adalah 5,38 detik dan pada saat percobaan dengan indeks (T3B) adalah 3,03 detik. Hasil kueri T3B 43,68% lebih cepat dari T3A.

Tabel 2. Hasil eksperimen

No	Activity	Avg(s)	Enhancement (%)
T1A	Query with Single Join, no index	5,66	
T1B	Query with Single Join, with index on Foreign Key	3.426	39.47
T2A	Query with Triple Join, with PK no FK	12,254	
T2B	Query with Triple Join, with PK, index PK and index FK	11.574	5.55
T3A	Query with Triple Join, where condition, no index	5.382	
T3B	Query with Triple Join, where condition, with index on Foreign Key	3.031	43.68

Perbandingan pengujian query pada penggabungan tabel dengan dan tanpa indeks dilakukan dengan jumlah tabel dan record data yang sama. Kinerja query diperoleh dengan indeks berfungsi dengan baik dalam mencocokkan baris yang relevan dari dua tabel atau empat tabel yang digunakan. Waktu eksekusi query lebih cepat dengan indeks, yaitu 3,42575; 11,57425 dan 3,03 detik. Ini menunjukkan bahwa penggunaan indeks mengoptimalkan operasi agregasi, mengurangi jumlah data yang akan diproses, dan meningkatkan efisiensi query. Ini karena MySQL dapat memanfaatkan indeks untuk mencocokkan baris yang relevan dalam gabungan, mengurangi jumlah data untuk diperiksa, dan meningkatkan efisiensi query.

Performa pada query yang tidak diindeks yang menggabungkan tabel bergantung pada jumlah baris yang cocok dengan kondisi JOIN, bukan pada penggunaan indeks. Waktu eksekusi kueri

lebih lambat tanpa indeks, pada 5,6; 12,254 dan 5,38 detik. Ini berarti tanpa indeks, MySQL harus memeriksa setiap baris di kedua tabel untuk mencocokkan baris yang sesuai, sehingga membutuhkan waktu lebih lama.

Dari hasil eksperimen didapatkan bahwa :

1. Indeks pada kolom yang digunakan dalam *sql join* dan pada kolom yang digunakan dalam pencarian memberikan peningkatan kinerja yang signifikan dalam operasi penggabungan tabel.
2. Dengan menggunakan indeks pada kolom *join* dan kolom pencarian, ditemukan bahwa waktu eksekusi *query table join* cenderung lebih cepat dibandingkan tanpa indeks.
3. Indeks memungkinkan MySQL mencocokkan baris yang relevan secara efisien, sehingga mengurangi jumlah data yang diperiksa, dan meningkatkan efisiensi query.
4. Tanpa indeks, MySQL harus memeriksa setiap baris di kedua tabel agar sesuai dengan baris yang sesuai, sehingga waktu eksekusi query menjadi lebih lama.
5. Memilih kolom yang tepat untuk diindeks dan memahami bagaimana indeks memengaruhi kinerja query sangat penting untuk desain database yang optimal.

Penting untuk dicatat bahwa hasil pengujian dapat bervariasi tergantung pada faktor lain, seperti kompleksitas agregasi, distribusi data, dan statistik tabel. Selain itu, hasil pengujian dapat berbeda jika penggabungan dilakukan pada kolom yang tidak diindeks atau jika indeks yang ada tidak cocok dengan *query*. Oleh karena itu, penting untuk melakukan pengujian representatif dan menyesuaikan strategi indeks yang sesuai dengan skema *database* dan *query* yang dijalankan.

SIMPULAN

Berdasarkan hasil percobaan dan analisis yang dilakukan, dapat disimpulkan bahwa penggunaan indeks dalam operasi *join* pada *query SQL*, baik pada tabel join 2 maupun tabel *join* 4, dengan menggunakan indeks pada kolom yang diindeks dapat memberikan peningkatan kinerja yang signifikan dalam operasi penggabungan tabel. Dengan adanya indeks, waktu eksekusi *query table join* cenderung lebih cepat dibandingkan tanpa indeks.

Efektivitas penggunaan indeks bergantung pada struktur tabel, distribusi data, dan jenis operasi gabungan yang dilakukan.

Dalam pengembangan aplikasi atau perancangan basis data perlu mempertimbangkan penggunaan indeks secara bijak. Indeks dapat menjadi alat yang ampuh untuk meningkatkan kinerja kueri, tetapi penggunaan indeks yang tidak sesuai atau tidak relevan dapat mengurangi kinerja secara keseluruhan. Oleh karena itu, perlu dilakukan pengujian yang cermat, mengoptimalkan struktur tabel, dan merancang indeks yang sesuai untuk memastikan kinerja yang optimal dalam operasi *JOIN* tabel.

SARAN

Pengujian yang dilakukan oleh peneliti dilakukan pada eksekusi *query* oleh satu user atau satu sesi user. Pada aplikasi yang sedang berjalan (mode *production* / *live*), sebuah tabel transaksi diakses oleh banyak user secara bersamaan, dan dapat dipastikan terjadi akses *query* dari banyak pengguna secara bersamaan. Perlu ada pengujian lebih lanjut penggunaan indeks pada multi transaksi *query*. Pada penelitian lanjutan dapat dilakukan pengujian terhadap peningkatan kinerja pada pemanfaatan partisi tabel di mySQL.

DAFTAR PUSTAKA

[1] D. V Dum, D. R. Zmaranda, C. A. Gy, L. Bandici, and D. E. Popescu, “applied sciences Performance Impact of Optimization Methods on MySQL Document-Based and Relational Databases,” 2021.

[2] P. Kieseberg, S. Schrittwieser, and E. Weippl, “Analysis of the Internals of MySQL / InnoDB B + Tree Index Navigation from a

Forensic Perspective," pp. 46–51, 2019, doi: 10.1109/ICSSA48308.2019.00013.

[3] D. V Dum, C. A. Gy, and S. Robert, "applied sciences Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application ' s Data Storage".

[4] S. Robert, "applied sciences An Analysis of the Performance and Configuration Features of MySQL Document Store and Elasticsearch as an Alternative Backend in a Data Replication Solution," 2021.

[5] T. Kraska *et al.*, "SageDB: A learned database system," *CIDR 2019 - 9th Biennial Conference on Innovative Data Systems Research*, 2019.

[6] B. Rahayudi, N. D. Priandani, B. T. Hanggara, and W. F. Mahmudy, "Database optimization for improved system performance and response time of hospital management information system," *Bulletin of Social Informatics Theory and Application*, vol. 5, no. 2, pp. 115–123, Sep. 2021.

[7] A. Anand, S. Das, O. Singh, and V. Kumar, "Testing resource allocation for software with multiple versions," *International Journal of Applied Management Science*, vol. 14, no. 1, p. 23, 2022, doi: 10.1504/IJAMS.2022.121040.

[8] A. de M. Del Esposte *et al.*, "Design and evaluation of a scalable smart city software platform with large-scale simulations," *Future Generation Computer Systems*, vol. 93, pp. 427–441, Apr. 2019, doi: 10.1016/j.future.2018.10.026.

[9] C. Li, J. Bai, and J. Tang, "Joint optimization of data placement and scheduling for improving user experience in edge computing," *J Parallel Distrib Comput*, vol. 125, pp. 93–105, Mar. 2019, doi: 10.1016/j.jpdc.2018.11.006.

[10] S. Maesaroh, H. Gunawan, A. Lestari, M. S. A. Tsaurie, and M. Fauji, "Query Optimization In MySQL Database Using Index," *International Journal of Cyber and IT Service Management*, vol. 2, no. 2, pp. 104–110, 2022, doi: 10.34306/ijcitsm.v2i2.84.

[11] P. Filip, "Comparison of MySQL and MongoDB with focus on performance," 2021.

[12] V. Nathan, J. Ding, M. Alizadeh, and T. Kraska, "Learning Multi-Dimensional Indexes," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 985–1000, 2020, doi: 10.1145/3318464.3380579.

[13] S. Sukirno and H. Suhendar, "Pengembangan Sistem Point of Sale Menggunakan Framework Codeignetier Berbasis Web," *Jurnal Algoritma*, vol. 19, no. 2, pp. 660–668, 2022, doi: 10.33364/algoritma/v.19-2.1181.

[14] S. Palanisamy and P. Suvithavani, "A survey on RDBMS and NoSQL Databases MySQL vs MongoDB," *2020 International Conference on Computer Communication and Informatics, ICCCI 2020*, 2020, doi: 10.1109/ICCCI48352.2020.9104047.

[15] M. Krommyda and V. Kantere, "Spatial Data Management in IoT systems: A study of available storage and indexing solutions," in *2020 Second International Conference on Transdisciplinary AI (TransAI)*, IEEE, Sep. 2020, pp. 146–153. doi: 10.1109/TransAI49837.2020.00033.

[16] R. Bardestani, G. S. Patience, and S. Kaliaguine, "Experimental methods in chemical engineering: specific surface area and pore size distribution measurements—BET, BJH, and DFT," *Can J Chem Eng*, vol. 97, no. 11, pp. 2781–2791, Nov. 2019, doi: 10.1002/cjce.23632.