ANALISIS PENGARUH IMPLEMENTASI *INDEX B-TREE* TERHADAP KINERJA WAKTU *DATABASE*

ANALYSIS OF THE EFFECT OF B-TREE INDEX IMPLEMENTATION ON DATABASE PERFORMANCE

Dita Ayu Eka Saputri[1*], Nimas Adjeng Nutfa Rabbaani[1], Putri Dia Lestari[1], Siti Mukaromah[1]
*E-mail : ditaputri1514@gmail.com

[1]Sistem Informasi, Ilmu Komputer, UPN "Veteran" Jawa Timur

Abstrak

Index B-Tree adalah struktur data yang digunakan dalam sistem basis data. Pengimplementasiannya untuk meningkatkan kinerja pencarian dan pengurutan data. Dilakukannya penelitian ini bertujuan untuk menganalisis pengaruh penggunaan index dengan konsep B-Tree terhadap kinerja waktu database. Salah satu teknik yang dapat digunakan adalah dengan mengimplementasikan *index B-Tree* dalam basis data. Maka dari itu, metode penelitian yang digunakan adalah metode eksperimental. Pengujian serangkaian query yang mencangkup operasi pencarian dan pengurutan data dieksekusi dengan dan tanpa implementasi *index B-Tree*. Waktu eksekusi query akan diukur dan dibandingkan antara kedua kasus tersebut. Dari pengujian, diharapkan dapat menunjukkan hasil bahwa *index B-Tree* memiliki pengaruh signifikan terhadap kinerja waktu database. Selain itu, penelitian ini juga dapat memberikan wawasan yang berguna untuk mencapai kinerja waktu yang optimal.
Kata kunci: *index, b-tree, kinerja, waktu, database.*

Abstract

Index B-Tree is a data structure used in database systems. Implementation to improve data search and sort performance. The purpose of this study was to analyze the effect of using an index with the B-Tree concept on database time performance. One technique that can be used is to implement a B-Tree index in the database. Therefore, the research method used is an experimental method. Testing a series of queries that include search operations and data sorting are executed with and without B-Tree index implementation. Query execution time will be measured and compared between the two cases. From the test, it is expected to show the results that the B-Tree index has a significant effect on database time performance. In addition, this research can also provide useful insights to achieve optimal time performance.
Keywords: *index, b-tree, performance, time, database.*

1. PENDAHULUAN

Di zaman digital yang terus berkembang, pengelolaan dan manajemen data yang efisien sangat penting dalam sistem basis data. Performa waktu yang optimal dalam basis data menjadi faktor penentu keberhasilan suatu aplikasi atau layanan. Seiring dengan itu, diperlukan pendekatan yang efektif untuk meningkatkan kinerja waktu dalam operasi-operasi database, seperti pencarian dan pengurutan data.

Penggunaan *index* pada *database* akan mempercepat kinerja dalam pengelolaan data, salah satunya menggunakan metode *index B-Tree*, *B-Tree* sendiri merupakan sebuah pohon pencarian dalam basis data yang dimana strukturnya memungkinkan data yang disimpan untuk melakukan *search*, *insert*, *delete* yang dapat dilakukan secara terstruktur[1]. *B-tree*

sendiri dibuat untuk memungkinkan dapat dilakukan penyimpanan banyak data dalam satu node, jumlah sub pohonnya juga dapat sangat banyak yang menyebabkan *B-tree* sangat cocok untuk digunakan dalam pengelolaan data[2].

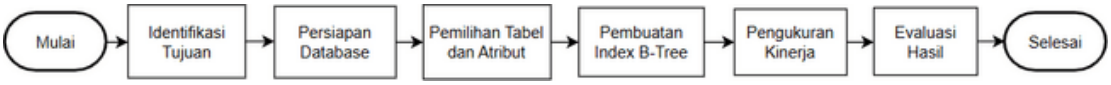
Index B-Tree adalah teknik yang terbukti efektif dalam mempercepat operasi pencarian dan pengurutan data. *Index B-Tree* digunakan untuk meningkatkan efisiensi waktu dalam basis data. Struktur data yang ditawarkan oleh *Index B-Tree* sangat efektif dalam melakukan pencarian dan pengurutan data sehingga memudahkan akses ke data yang dibutuhkan dengan cepat. *Index* pada database digunakan untuk mencari nilai kolom pada tabel tertentu dengan cepat karena tanpa menggunakan *index* maka *database* harus melakukan pencarian dari mulai tabel baris pertama hingga tabel baris terakhir dan akan memakan banyak waktu[3]. Selain itu, *B-tree* memastikan data tetap terurut dan seimbang, sehingga operasi pengurutan data dapat dilakukan dengan efisien.

Analisis kinerja waktu *database* adalah proses penting dalam pengelolaan dan pengoptimalan sistem *database*. Kinerja waktu mengacu pada waktu yang dibutuhkan sistem untuk melakukan pengambilan data, pembaruan atau operasi operasi pada *database*. Kualitas kinerja waktu yang baik sangat penting untuk memastikan daya tanggap, efisiensi, dan skalabilitas sistem *database*. Penggunaan *index* pada *database* memangkas waktu yang digunakan menjadi lebih cepat dalam proses mencari nilai kolom pada tabel tertentu[4].

Dalam penelitian ini akan mengetahui dampak dari implementasi pengindeksan B-Tree pada kinerja waktu *database*. Fokus utama pada penelitian ini untuk mengukur waktu yang diperlukan untuk operasi pencarian dan pembaruan data, baik dengan *index B-Tree* maupun tanpa *index B-Tree*. Melalui analisis ini, kita dapat mengevaluasi apakah implementasi index B-Tree memberikan peningkatan dalam kinerja waktu database. Penelitian ini bertujuan untuk memberikan pemahaman yang lebih baik mengenai pentingnya *index B-Tree* dalam meningkatkan kinerja waktu *database* dan dampak implementasi *index* B-Tree dapat mempengaruhi kinerja waktu *database*.

2. METODOLOGI

Metode penelitian yang digunakan adalah studi literatur dan difokuskan kepada metode eksperimental. Metode eksperimental adalah pendekatan yang digunakan untuk mengevaluasi hubungan sebab-akibat antara variabel yang dapat memberikan memberikan pemahaman tentang pengaruh implementasi *index B-Tree* terhadap kinerja waktu *database*. Metode eksperimental memungkinkan variabel dimanipulasi dan diubah sesuai kebutuhan[5]. Tahapan penelitian menggunakan metode eksperimental untuk implementasi *index B-Tree* dalam *database oracle* ditunjukkan pada Gambar 1.



Gambar 1. Alur Pengujian Implementasi Index

2.1 Identifikasi Tujuan

Pada penelitian ini, bertujuan untuk mengevaluasi apakah penggunaan *index B-Tree* dapat meningkatkan kinerja waktu dalam operasi *database* atau pengeksekusian *query* dalam *database*, seperti mengidentifikasi perbedaan kinerja waktu antara *database* yang menggunakan*index B-Tree* dan yang tidak menggunakan *index B-Tree*.

2.2 Persiapan Database

Dengan mempersiapkan *database* yang tepat, penelitian dapat dilakukan untuk memastikan tabel relevan dan data yang relevan sesuai kondisi untuk menganalisis pengaruh implementasi *index B-Tree* terhadap kinerja waktu *database*. Pemilihan *database* yang mendukung *index B-Tree* dan memiliki fitur yang memungkinkan pengukuran kinerja waktu.

2.3 Pemilihan Tabel dan Atribut

Penelitian fokus pada pencarian data, tabel utama yang berisi data yang akan dicari dapat

dipilih dengan memastikan adanya tabel yang relevan terdapat atribut yang dapat di *index* menggunakan *index B-Tree* dan tabel yang dipilih mewakili jumlah data yang cukup besar.

2.4 Pembuatan Index B-Tree

Salah satu aspek penting dalam penelitian ini yang digunakan untuk meningkatkan kinerja waktu operasi *database*. Identifikasi atribut-atribut dengan tujuan atribut yang sering digunakan dalam operasi *database* dapat memiliki potensi besar untuk meningkatkan kinerja waktu. Pada implementasikan *index B-Tree* ke dalam *database* dan atribut yang dipilih menggunakan perintah CREATE INDEX untuk membuat *index* yang sesuai. Penggunaan *index* yang tepat bergantung pada jenis nilai yang terdapat dalam kolom yang akan di *index*[5].

Pembuatan *index* dilanjutkan dengan penamaan *index*. Ada berbagai cara memberikan nama
index:

1. Berakhiran IDX, contohnya EMPLOYEES_IDX
2. Berawalan IDX, contohanya IDX_EMPLOYEES
3. Berawalan I, contohnya I_EMPLOYEES
4. Berakhiran nama kolom, contohnya EMPLOYEES_FIRST_NAME_FK
5. Berawalan I dan akhiran sequence, contohnya I_EMPLOYEES_01 danI_EMPLOYEES_02.

Membuang *index* bisa dilakukan secara otomatis dan manual, sama seperti membuat *index*. Jika *index* dibuang secara manual, digunakan *query* DROP INDEX atau *index* akan dibuang secara otomatis jika tabel asalnya dibuang[9].

2.5 Pengukuran Kinerja

Pada pengukuran kinerja ini tujuannya adalah membandingkan kinerja waktu operasi *database* sebelum implementasi *index B-Tree* (tanpa *index B-Tree*) dan kondisi setelah implementasi *index B-Tree*. Dari *query* yang dijelaskan, parameter yang digunakan untuk menguji perintah dalam pencarian data adalah kecepatan waktu mengeksekusi dan menampilkan datanya[7].

2.6 Evaluasi Hasil

Pada tahap ini, melakukan identifikasi dampak pengaruh implementasi *index B-Tree* terhadap kinerja waktu *database*. Evaluasi waktu eksekusi dapat diperhatikan pada peningkatan atau penurunan waktu eksekusi yang terjadi setelah implementasi *index B-Tree*. Dalam penelitian lain yang serupa, memperoleh hasil bahwa penggunaan *index* mampu mengoptimalkan proses pencarian dan penghitungan data. Optimal dalam artian waktu respon setelah diberi *index* lebih cepat dibanding sebelum diberi[8].

3. HASIL DAN PEMBAHASAN

Implementasian *index database* perlu perancangan dan pertimbangan strategi yang tepat karena tidak semua tabel perlu menggunakan *index*. Berikut adalah pemaparan dari hasil dan pembahasan dari pengaruh implementasi *index B-tree* terhadap kinerja waktu *database*:

3.1 Manfaat *Index* B-Tree

Index B-tree memungkinkan akses dan pengurutan data hanya pada bagian yang relevan, bukan seluruh dataset. Dengan struktur indeks hierarkis, pencarian dan penyortiran data dapat dilakukan dengan kompleksitas waktu yang lebih rendah daripada mencari secara linier melalui seluruh data. Ini mengurangi kompleksitas waktu untuk pengurutan data, menghasilkan pencarian dan penyortiran yang lebih cepat, serta meningkatkan responsivitas sistem.

3.2 Waktu Pencarian Data

Index B-tree mengorganisir data secara hierarkis, dengan tingkat paling atas mengarahkan ke blok-blok data yang lebih spesifik. Pada saat pencarian data, *index B-tree* memungkinkan sistemuntuk menghindari blok-blok data yang tidak relevan dan hanya fokus pada blok-blok

yang berpotensi mengandung data yang dicari. Hal ini mengurangi kompleksitas waktu yang dibutuhkan untuk pencarian data, dibandingkan dengan metode pencarian linear yang harus memeriksa setiap baris data secara berurutan. Dengan mengurangi kompleksitas waktu, *index B-tree* mempercepat waktu pencarian data. Oleh karena itu, waktu pencarian menjadi lebih cepat, sehingga sistem menjadi lebih responsif terhadap permintaan.

3.3 Menentukan Metode Pembuatan Index

Implementasi *index* dapat terlaksana di dalam sebuah tabel untuk mencari data lebih cepat dan efisien. Untuk pembuatan *database*, menggunakan *query* berikut:

CREATE INDEX *index_name*
ON *table_name* (*column_name*); … (1)

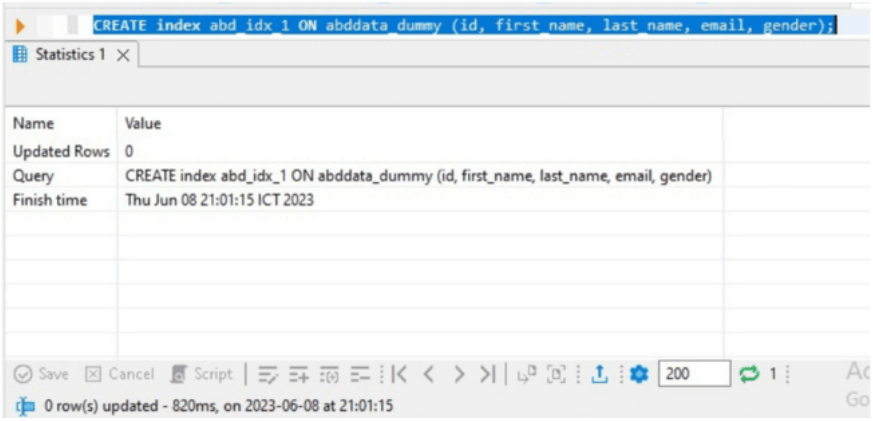
Query (1) digunakan untuk membuat *index* dalam tabel dengan kolom tertentu. Kemudian untuk menghapus *index* yang sudah ada menggunakan DROP INDEX, seperti contoh *query* berikut[10].

DROP INDEX *index_name*
ON *table_name;* …(2)

Query (2) ditujukan untuk menghapus *index* yang sudah ada atau sudah dibuat melalui *DROP INDEX* dan dilanjutkan dengan nama *index* yang akan dihapus di dalam nama tabel yang akan dihapus.

3.4 Implementasi Pengujian Penggunaan Index B-Tree

Pengujian implementasi *index B-Tree* dilakukan di aplikasi DBeaver dengan menyambungkan localhost MySQL. Untuk langkah pertama adalah menyediakan sebuah *database* yang akan diuji implementasian *index B-Tree*. Setelah *database* tersedia, dibuat *index* di kolom dalam tabel yang dipilih.



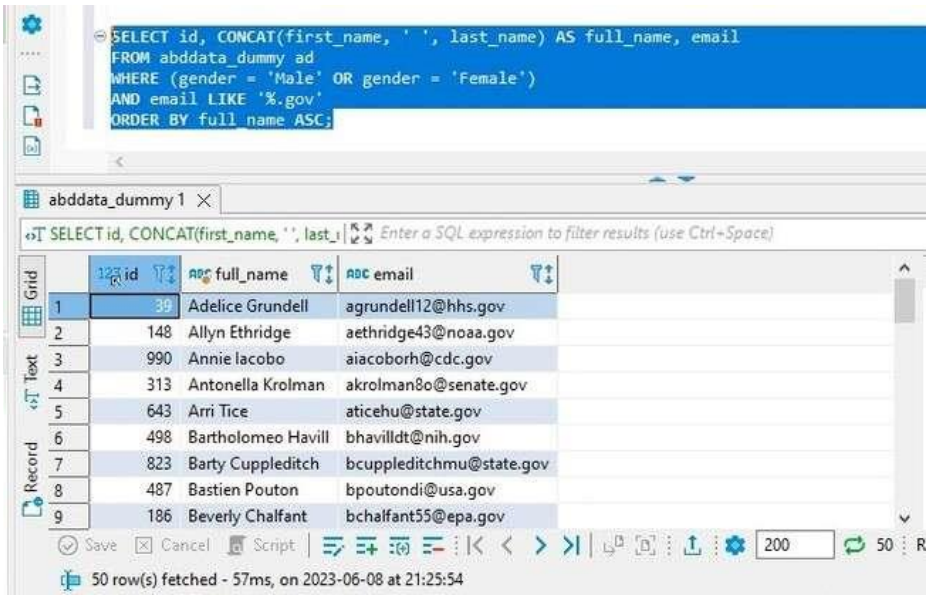
Gambar 2. Pembuatan *Index*

Selanjutnya melakukan pengujian yang ketiga untuk membandingkan apabila menggunakan dantidak menggunakan *index*. Berikut *query* yang digunakan.

SELECT *nama_kolom*, CONCAT(*nama_kolom*, ' ', *nama_kolom*) AS *nama_kolom_baru*, *nama_kolom* FROM *nama_tabel* WHERE (*nama_kolom* = 'value' OR *nama_kolom*= 'value') AND *nama_kolom* LIKE '%example' ORDER BY *nama_kolom* ASC;

SELECT id, CONCAT(first_name, ' ', last_name) AS full_name, email FROM abddata_dummy ad WHERE (gender = 'Male' OR gender = 'Female') AND email LIKE '%.gov' ORDER BY full_name ASC;

Hasil dari pengujian menunjukkan waktu pengeksekusian selama 57 ms sesuai Gambar 5.



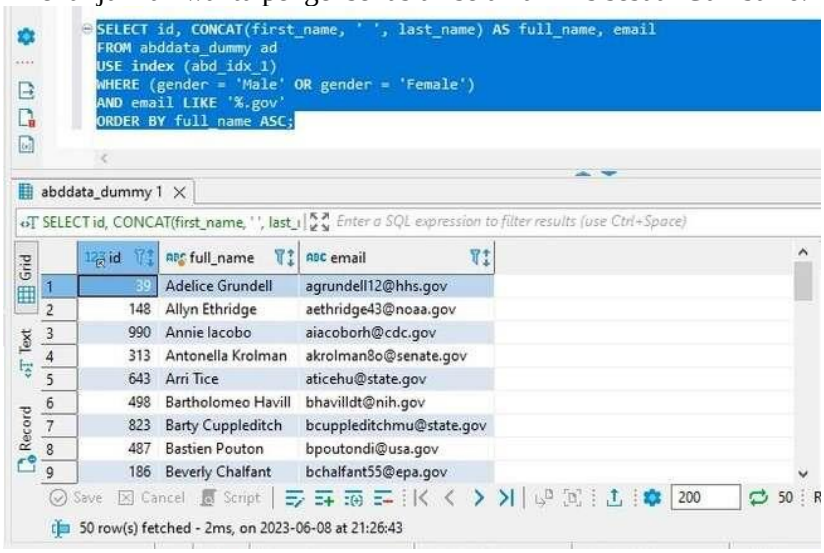
Gambar 5. Pengujian Tanpa *Index (2)*

Lalu pengujian keempat dilakukan menggunakan *index* yang dibuat sendiri menggunakan *query* sebagai berikut.

SELECT *nama_kolom*, CONCAT(*nama_kolom*, ' ', *nama_kolom*) AS *nama_kolom_baru*, *nama_kolom* FROM *nama_tabel* USE index (*nama_index*) WHERE (*nama_kolom* = 'value' OR *nama_kolom*= 'value') AND *nama_kolom* LIKE '%example' ORDER BY *nama_kolom* ASC;

SELECT id, CONCAT(first_name, ' ', last_name) AS full_name, email FROM abddata_dummy ad USE index (abd_idx_1) WHERE (gender = 'Male' OR gender = 'Female') AND email LIKE '%.gov' ORDER BY full_name ASC;

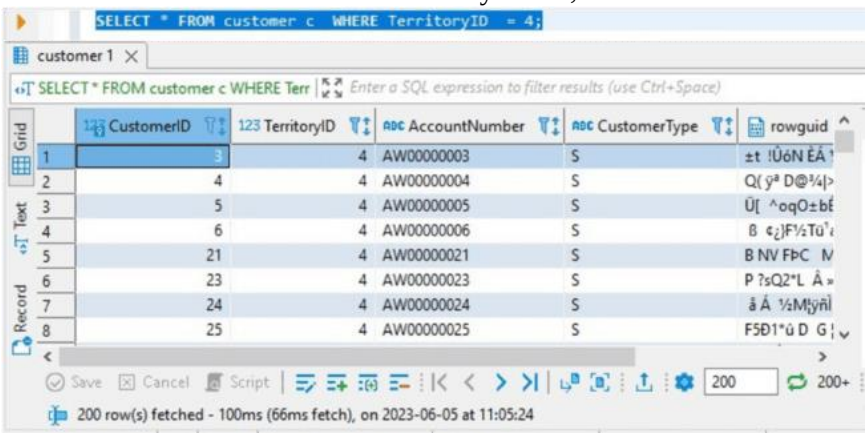
Hasil dari pengujian menunjukkan waktu pengeksekusian selama 2 ms sesuai Gambar 6.



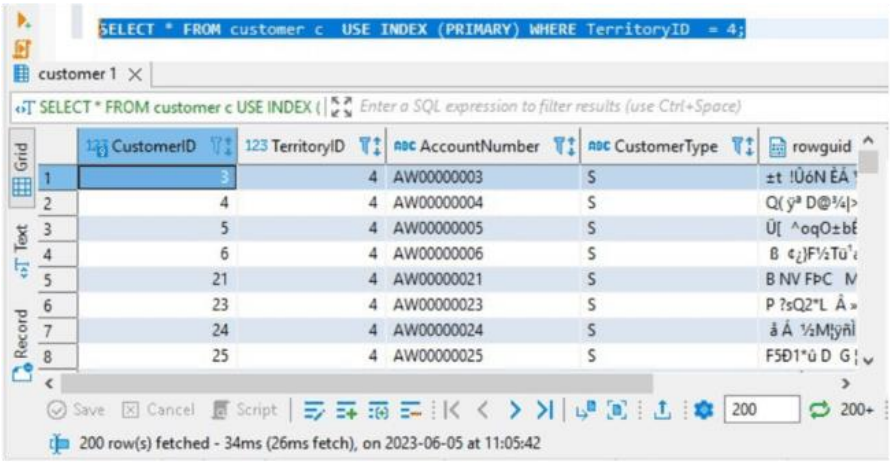
Gambar 6. Pengujian Menggunakan *Index (2)*

Selain menguji dengan membuat *index* sendiri, pengimplementasian *index* diuji menggunakan *database* yang sudah mempunya *index B-Tree* bawaan. Pada gambar di bawah menunjukkan pengujian yang tidak menggunakan *index* dengan *query* sebagai berikut.

SELECT * FROM customer c WHERE TerritoryID = 4;



Gambar 7. Pengujian Tanpa *Index* Tersedia



Gambar 8. Pengujian dengan *Index* Tersedia

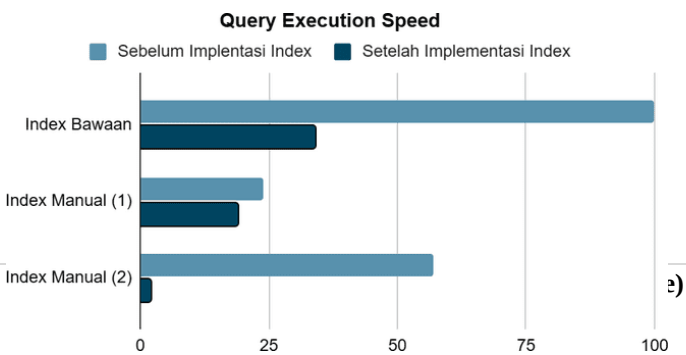
3.6 Perbandingan Hasil Pengujian

Dari hasil pengujian, didapatkan hasil yang dapat dibandingkan pada Tabel 1.

Tabel 1. Tabel Perbandingan Hasil Pengujian

Kondisi	Sebelum Implementasi Index	Setelah ImplementasiIndex
Penggunaan Index Bawaan	100 ms (66 ms fetch)	34 ms (26 ms fetch)
Penggunaan Index Manual (1)	24 ms (2 ms fetch)	19 ms (1 ms fetch)
Penggunaan Index Manual (2)	57 ms	2 ms

Pada hasil analisis data dari pengujian tersebut akan divisualisasikan hasilnya menggunakan grafik. Grafik perbandingan rata-rata kinerja waktu pada *database* ditunjukkan pada Gambar 9.



Gambar 9. Grafik perbandingan waktu eksekusi *query index B-Tree*

4. KESIMPULAN DAN SARAN

Berdasarkan analisis pengaruh terhadap analisis penerapan *index B-Tree* terhadap kinerja waktu *database*, dapat disimpulkan bahwa penggunaan *index B-Tree* pada *database* dapat mempercepat dalam hal kinerja waktu dengan percobaan 3 kali pengujian dengan kondisi menggunakan *index* manual dan mengeksekusi query 66 ms lebih cepat. Pengujian kedua menggunakan *index* manual dan mengeksekusi query 5 ms dan 55 ms lebih cepat. Dengan penggunaan *index B-Tree* pada *database*, dapat dilakukan dengan menjalankan *query* pada *index* yang berdampak lebih cepat dan mengurangi waktu akses data. Penggunaan *index* sangat penting pada *database* yang besar dan kompleks, dan pada *query* yang sering dilakukan pada kolom yang sering dicari atau digunakan sebagai kriteria pencarian.
Oleh karena itu, pada kolom yang akan di*index* harus dipilih dengan hati-hati, dengan mempertimbangkan *query* yang sering dieksekusi. Saran penelitian selanjutnya dapat berfokus pada membandingkan implementasi *index B-Tree* dengan metode *index bitmap* dan dapat memberikan lebih banyak pemahaman dari setiap metode pengindeksan, serta kinerja waktu yang dihasilkan.

5. DAFTAR PUSTAKA

[1] Diva Putri Anasya, 2022, “Optimasi Hasil Query Pada Fitur Pencarian Platform Marketplace Penjualan dan Sewa Menyewa Buku Online”, *Seminar Nasional & Call Paper Fakultas Sains dan Teknologi(SENASAINS)*, Sidoarjo, 2 Juni.
[2] Dike Pradana Putra, 2015, “Implementasi Full Text Indexing pada Dokumen Elektronik dengan Algoritma B-Tree”, *e-Proceeding of Engineering*, Vol 2, hal 1119.
[3] Raharjo, Suwanto, "Integrity Constraint Basis Data Relasional Dengan Menggunakan pl/pgsql Dan Check Constraint " in *Seminar Nasional Teknik dan Manajemen Industri, Malang*, 2015, pp. IV-22 - IV 27.
[4] R. Pamungkas, 2018, “Optimalisasi Query Dalam Basis Data Mysql Menggunakan Index”, Res. Comput. Inf. Syst. Technol. Manag., vol. 1, no. 1, hal. 27.
[5] Ermatita, E. Analisis Optimasi Query Pada Data Mining. *Sriwijaya Journal of Information Systems, 1*(1), 130026.
[6] Hutabarat, I.B., 2004. *ORACLE 9i DBA.* Yogyakarta: Penerbit ANDI.
[7] Junus Sinuraya, 2017, “Metode Pencarian Data Menggunakan Query Hash Join Dan QueryNested Join”, Jurnal Teknovasi, vol. 04, no. 01, p. 44.
[8] Bernardo, J.F.A., dkk, 2018, “Impelementasi Penerapan Index Dalam Mempercepat Eksekusi Query Pada Basis Data”, hal. 10.
[9] R. Pamungkas, 2018, “Optimalisasi Query Dalam Basis Data Mysql Menggunakan Index”, Res. Comput. Inf. Syst. Technol. Manag., vol. 1, no. 2, hal. 27.
[10] X-Oerang Technology., 2004. Pemrograman Menggunakan Oracle Developer Tingkat Lanjut Edisi 1, Indonesia : Yogyakarta.