Aplikasi *B-Tree* dalam Sistem Basis Data

I Putu Bakta Hari Sudewa - 13521150[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521150@std.stei.itb.ac.id

***Abstract*— Pada era ini yang tiap harinya tak terlepas dari teknologi dan internet, akses akan informasi dari mana saja dan kapan saja menjadi hal yang sangat penting untuk semua orang. Data-data yang menjadi sumber informasi, sarana hiburan dan sebagainya tentu saja merupakan hal yang sangat krusial dan harus dapat tersalurkan dengan baik. Basis data merupakan salah satu faktor kunci dalam menyimpan dan menyalurkan data-data secara efisien dan optimal. Dengan adanya mekanisme indeks pada basis data, hal ini bukan lagi menjadi masalah. Didasari dengan struktur data *B-Tree*, indeks pada basis data memperkecil jumlah operasi yang diperlukan dalam proses pengolahan suatu data.**

***Kata Kunci*—pohon, basis data, indeks.**

I. PENDAHULUAN

Data merupakan suatu hal yang tak dapat terlepas dari kehidupan kita. Setiap aspek yang ada di dunia ini merupakan suatu data yang diolah dan direpresentasikan dengan cara tertentu. Contohnya, seperti *smartphone* ataupun laptop yang kita selalu gunakan tiap harinya merupakan suatu kumpulan data-data yang diolah sedemikan rupa sehingga dapat kita gunakan sebagai sumber informasi, membantu pekerjaan, sarana hiburan, dan sebagainya.

Jika ada data, tentunya ada suatu tempat atau alat untuk menyimpan data tersebut. Misalnya, seperti otak kita yang menyimpan berbagai memori tentang peristiwa-peristiwa yang pernah kita alami dapat dikatakan sebagai alat penyimpanan data. Untuk data-data lainnya seperti kumpulan foto-foto yang diunggah di media sosial, video yang kita tonton di media *streaming*, jadwal penerbangan, dan sebagainya juga disimpan dalam suatu penyimpanan yang disebut dengan basis data.

Penyimpanan data-data digital ini dalam suatu basis data telah dimulai sejak tahun 1960 dan pada tahun 1970 diusulkan suatu representasi baru dalam basis data yang kita gunakan hingga saat ini, yaitu model data relasional. Model data relasional menggunakan tabel dua dimensi, yang terdiri atas baris dan kolom untuk menggambarkan sebuah dokumen data.

Selain model data relasional yang menggunakan tabel, saat ini juga terdapat berbagai macam model-model lainnya seperti model nonrelasional yang tidak memiliki bentuk khusus (fleksibel). Terlepas dari berbagai macam model basis data yang ada, terdapat beberapa hal fundamental yang digunakan di seluruh model basis data, yaitu terkait pembuatan, pembacaan, pembaruan, dan penghapusan suatu data dari basis data.

Seiring dengan banyaknya data yang disimpan oleh suatu basis data, proses pembacaan suatu data akan membutuhkan sumber daya yang lebih besar dan memerlukan waktu yang lebih lama. Untuk mengatasi masalah performa dalam proses pembacaan data ini, terdapat suatu motode yang disebut dengan indeks (*indexing*).

Indeks disini sangat mirip dengan indeks yang umumnya terdapat di halaman belakang suatu buku, yaitu suatu struktur data baru yang dibuat dengan melakukan pengelompokan dan pengurutan tertentu terhadap data yang ada sehingga waktu yang dibutuhkan dalam pemrosesan suatu data akan berkurang. Implementasi dari indeks ini menggunakan suatu struktur data yang disebut dengan *B-Tree* atau pohon seimbang yang memiliki derajat dan aturan-aturan tertentu. Dengan adanya struktur data ini proses pembacaan data dapat dilakukan beberapa kali lipat lebih efisien dibanding sebelumnya.

II. LANDASAN TEORI

A. Basis Data

1) *Definisi Basis Data*

Basis data adalah sekumpulan data yang dikelola dengan sedemikan rupa berdasarkan ketentuan tertentu yang saling berkaitan sehingga memudahkan dalam pengelolaannya. Lewat pengelolaan itulah pengguna bisa mendapatkan kemudahan dalam mencari sebuah informasi, membuang informasi, maupun menyimpan informasi.

2) *Jenis dan Fungsi Basis Data*

a) *Basis Data Operasional*

Basis data operasional adalah basis data yang memiliki fungsi sebagai suatu tempat untuk mengelola data dinamis secara langsung. Basis data jenis ini memungkinkan para penggunanya untuk bisa melihat, melakukan, dan memodifikasi data. Modifikasi data yang dimaksud yakni dengan cara menambah atau mengubah, ataupun menghapus data secara langsung lewat suatu perangkat keras yang dipakai.

Contoh basis data operasional adalah *JSON* dan *XML*

b) *Basis Data Warehouse*

Basis data *warehouse* merupakan basis data yang dipakai untuk melakukan pelaporan dan analisis data. Basis data *warehouse* adalah sentral data terpadu yang berasal dari satu hingga lebih dari satu sumber yang berbeda.

Contoh basis data *Warehouse* adalah Microsoft SQL Server.

c) *Basis Data Terdistribusi*

Basis data terdistribusi adalah suatu basis data dengan perangkat penyimpanannya yang tidak terpasang pada sebuah perangkat komputer lainnya yang saling berkaitan. Berbeda dengan sistem paralel yang terhubung erat dan bersistem pada data tunggal. Basis data ini terdistribusi lewat suati situs yang tergabung dan tidak mempunyai sebuah kompenen fisik.

d) *Basis Data Relasional*

Basis data relasional adalah basis data yang mengorganisir berdasarkan pada model hubungan data. Terdapat banyak perangkat lunak yang memakai sistem ini untuk mengatur dan memelihara basis data melalui hubungan setiap data.

Contoh basis data relasional adalah MySQL, PostgreSQL, MariaDB, MongoDB, Oracle Database, SAP HANA, dan IBM Db2.

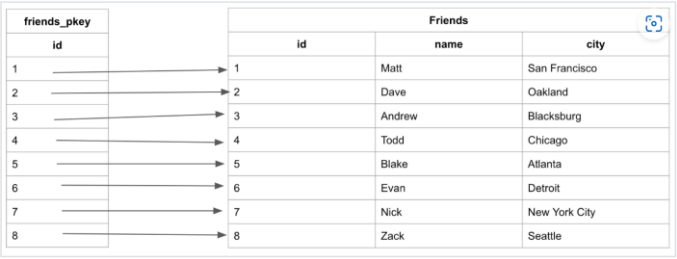
B. Indeks Basis Data

Indeks basis data adalah struktur data yang mengatur rekaman data pada *disk* untuk mengoptimalkan beberapa jenis operasi pengambilan tertentu. Indeks secara efektif mengambil semua rekaman yang memenuhi syarat pencarian pada kolom kunci pencarian.

Indeks tambahan dapat dibuat pada kumpulan rekaman data tertentu, dengan kunci pencarian yang berbeda, untuk mempercepat pencarian. Entri data adalah rekaman yang disimpan dalam file indeks. Pencarian indeks secara efisien untuk menemukan entri data yang diinginkan, lalu menggunakannya untuk mendapatkan rekaman data.

Pencarian data secara sekuensial terhadap rekaman pada tabel akan memakan waktu yang lama, maka pada dasarnya terdapat beberapa pengurutan, salah satunya adalah pengurutan secara indeks, yaitu berdasarkan urutan dari sebuah nilai.

Terdapat dua jenis indeks dalam basis data, yaitu indeks dengan kluster dan indeks tanpa kluster. Indeks dengan kluster adalah suatu indeks unik yang dimiliki oleh setiap tabel ketika tabel tersebut dibuat dengan menggunakan suatu *primary key* untuk mengatur tabel tersebut. Suatu indeks dengan kluster menjamin *primary key* yang ada pada tabel terurut membesar dan merupakan urutan yang disimpan pada memori.

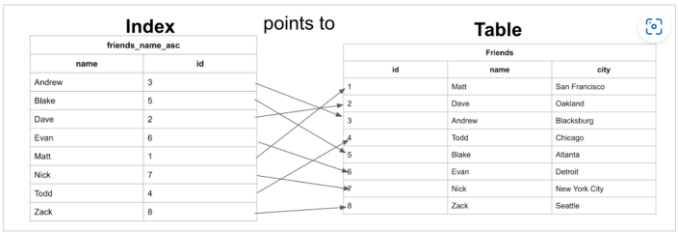


Gambar 1. Contoh indeks dengan kluster
(Sumber: https://dataschool.com/sql-optimization/how-indexing-works/)

Beberapa sifat dari indeks dengan kluster, antara lain

a) Tidak harus dideklarasikan secara eksplisit.
b) Dibuat bersamaan saat pembuatan tabel.
c) Menggunakan *primary key* yang terurut membesar.

Indeks tanpa kluster adalah indeks yang diurutkan berdasarkan kolom tertentu yang ada pada tabel. Indeks tanpa kluster dapat dibuat kapan saja setelah pembuatan tabel. Indeks tanpa kluster ini memiliki pointer yang mengarah kepada data yang tersimpan pada memori, sehingga proses *query* yang dilakukan nantinya dapat lebih efisien karena operasi yang dibutuhkan untuk mengakses data pada memori berkurang.

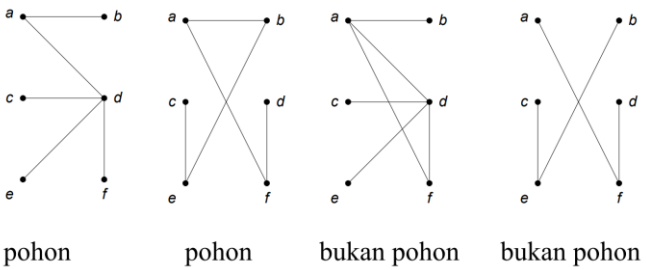


Gambar 2. Contoh indeks tanpa kluster
(Sumber: https://dataschool.com/sql-optimization/how-indexing-works/)

C. Pohon

1) Definisi Pohon

Pohon adalah graf tak berarah terhubungan yang tidak mengandung sirkuit. Sedangkan hutan adalah kumpulan pohon yang saling lepas atau graf tak terhubung yang tidak mengandung sirkuit. Setiap komponen di dalam graf terhubung tersebut adalah pohon.



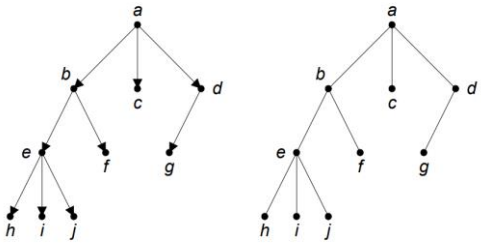
Gambar 3. Contoh pohon dan bukan pohon
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf)

2) Sifat-sifat Pohon

Misalkan $G = (V, E)$ adalah graf tak berarah sederhana dna jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:

a) G adalah pohon
b) Setiap pasang simpul di G terhubung dengan lintasan tunggal.
c) G terhubung dan memiliki $m = n - 1$ buah sisi.
d) G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi.
e) G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
f) G terhubung dan semua sisinya adalah jembatan.

3) *Pohon Berakar (rooted tree)*
Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah.



(a) Pohon berakar (b) sebagai perjanjian, tanda panah pada sisi dapat dibuang

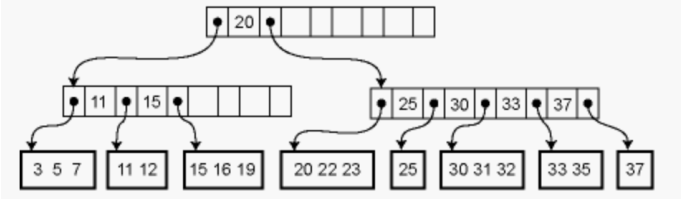
Gambar 4. Pohon berakar
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf)

D. B-Tree

B-Tree pada awalnya diciptakan untuk menyimpan data pada *disk*, saat lokalitas dianggap lebih penting dibandingkan dengan penggunaan memori. *B-Tree* dengan derajat m adalah sebuah pohon pencarian dengan jumlah maksimum anak untuk tiap simpul selain simpul daun sebanyak m buah.

B-Tree memiliki beberapa aturan atau sifat, yaitu

a) Setiap daun berada pada kedalaman yang sama.
b) Jika suatu simpul memiliki n buah anak, maka simpul tersebut mengandung $n - 1$ *key*.
c) Setiap simpul internal minimal memiliki $\left\lceil \frac{n}{2} \right\rceil$ anak.
d) Simpul akar minimal memiliki 2 buah anak jika tidak sekaligus menjadi daun.
e) Elemen-elemen yang tersimpan pada suatu upapohon dari *B-Tree* semunya memiliki nilai *key* di antara nilai *key* yang ada pada simpul asal.



Gambar 5. Contoh *B-Tree*
(Sumber: https://www.cs.cornell.edu/courses/cs3110/2012sp/recitations/rec25-B-trees/rec25.html)

III. ANALISIS DAN PEMBAHASAN

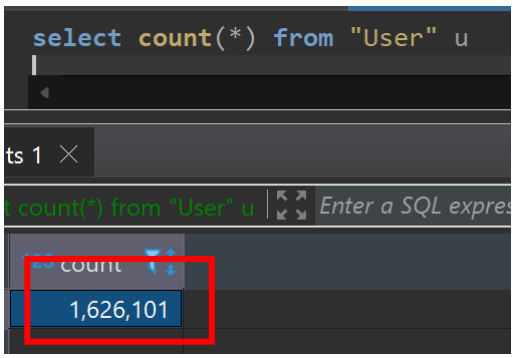
A. Proses Seeding pada Basis Data

Seeding adalah proses pengisian basis data dengan suatu data tertentu yang umumnya merupakan sebuah sampel atau data acak. Untuk analisis kali ini telah dibuat sebuah basis data bernama testing yang merupakan suatu basis data dengan model relasional. Pada basis data ini, terdapat sebuah tabel bernama User yang terdiri atas dua kolom, yaitu id dan username.

	id	username
1	11	user0
2	12	user1
3	13	user2
4	14	user3
5	15	user4
6	16	user5
7	17	user6

Gambar 6. Tabel user
(Sumber: Dokumen pribadi penulis)

Pada gambar 6, terlihat kolom id dimulai dari 11 dan kolom username dimulai dari user0. Untuk baris-baris berikutnya memiliki pola yang sama hingga id 1.626.111 dan username user1626100. Jumlah data dapat dilihat pada gambar 7, yaitu sebanyak 1.626.101 data.



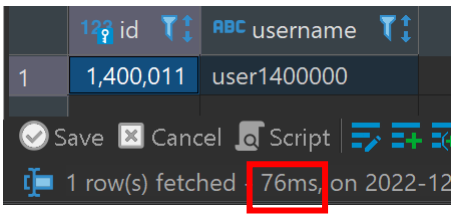
Gambar 7. Banyak data pada tabel user
(Sumber: Dokumen pribadi penulis)

B. Pembacaan Data Tanpa Indeks

Berikutnya akan dilakukan proses pembacaan suatu data dari tabel User dengan menggunakan *query* pada gambar 8. Basis data secara bawaan melakukan pembacaan data secara sekuensial. Proses pembacaan data tersebut memerlukan total waktu sebesar 76 ms. Total waktu yang diperoleh ini mungkin terlihat cukup cepat, namun untuk data dan jumlah kolom yang lebih besar, pembacaan tanpa indeks akan memerlukan lebih banyak waktu. Hal ini tentunya akan menurunkan performa proses-proses yang berhubungan dengan data tersebut.

```
select * from "User" u where u.username = 'user1400000'
```

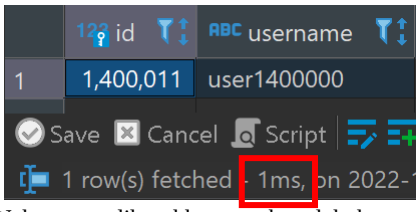
Gambar 8. *Query* untuk mendapatkan data User dengan username user1400000
(Sumber: Dokumen pribadi penulis)



Gambar 9. Waktu yang dibutuhkan untuk melakukan *query* tanpa indeks
(Sumber: Dokumen pribadi penulis)

C. Pembacaan Data dengan Indeks

Untuk mengatasi masalah performa akibat banyaknya data pada suatu tabel, disinilah indeks memiliki peran yang sangat penting. Selanjutnya, dengan menggunakan *query* yang sama seperti pada gambar 8, tetapi dengan penambahan indeks pada kolom username, diperoleh total waktu yang jauh lebih sedikit, yaitu 1 ms, 76x lebih cepat dari sebelumnya.



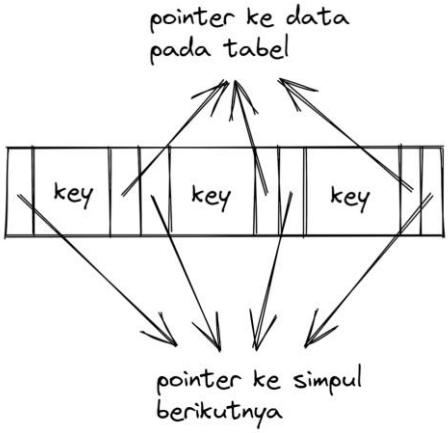
Gambar 10. Waktu yang dibutuhkan untuk melakukan *query* dengan indeks
(Sumber: Dokumen pribadi penulis)

D. Implementasi B-Tree pada Indeks Sistem Data

Peningkatan performa yang signifikan akibat menggunakan indeks dapat terjadi karena digunakannya suatu struktur data, yaitu *B-Tree* untuk memotong total operasi yang harus dilakukan untuk memperoleh atau membaca data yang diinginkan.

Misalnya akan diambil 10 data dari tabel User pada gambar 6. Indeks akan dilakukan pada kolom username, namun dalam analisis kali ini, username akan direpresentasikan dengan bilangan bulat mulai dari 1 hingga 10.

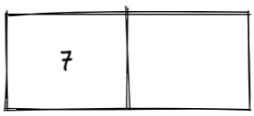
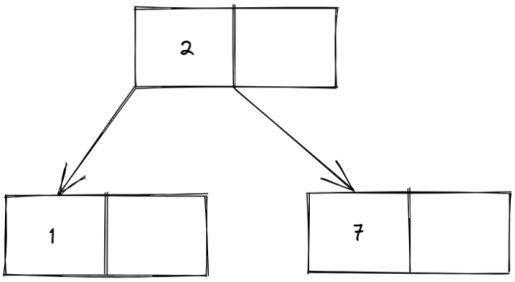
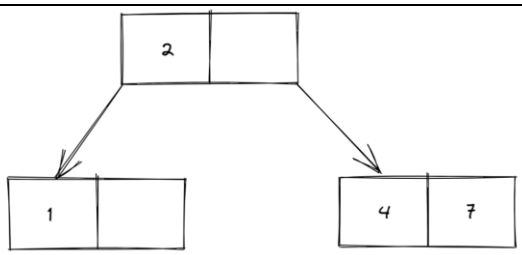
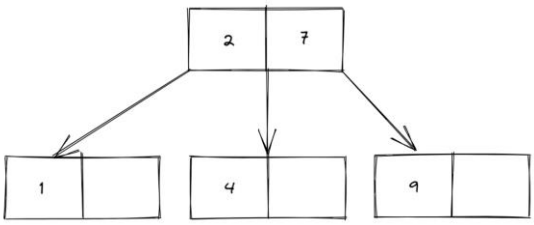
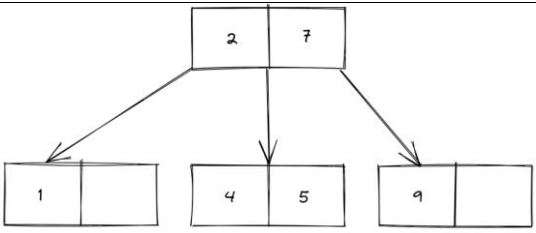
Ketika *query* indeks dijalankan maka akan dibentuk sebuah *B-Tree*. Masing-masing simpul pada *B-Tree* yang terbentuk memiliki struktur seperti pada gambar 11.

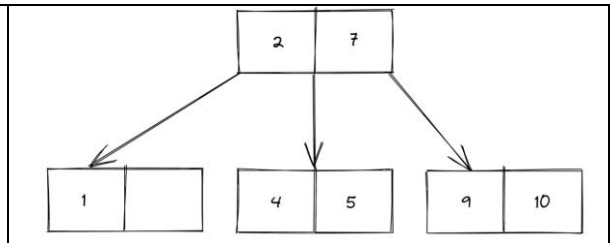
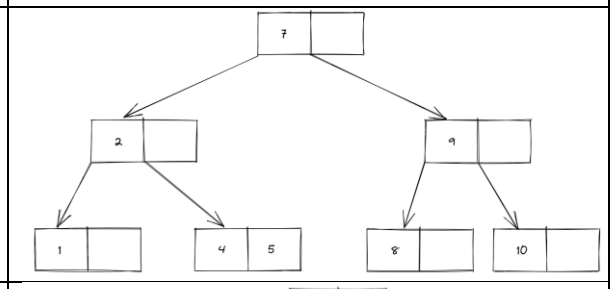
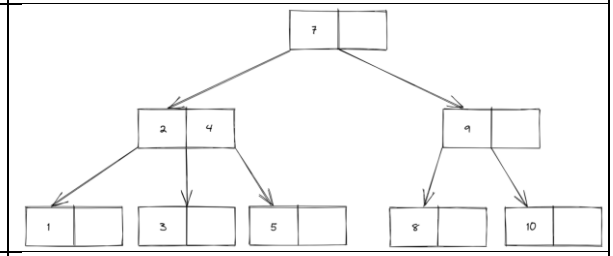
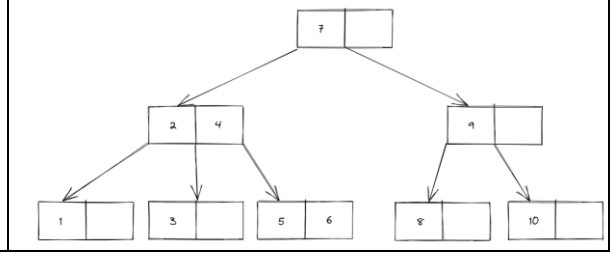


Gambar 11. Representasi simpul pada *B-Tree*
(Sumber: Dokumen pribadi penulis)

Untuk analisis kali ini diasumsikan *B-Tree* tersebut memiliki derajat $n = 3$, sehingga sesuai dengan syarat untuk membentuk suatu *B-Tree*, banyaknya *key* maksimal pada setiap simpul adalah $n - 1 = 2$ dan banyaknya anak minimal untuk masing-masing simpul internal (simpul yang bukan akar atau daun) adalah $\left\lceil \frac{n}{2} \right\rceil = 2$.

Selanjutnya, akan dilakukan penambahan simpul secara sekuensial. Dengan urutan 7, 2, 1, 4, 9, 5, 10, 8, 3, 6, proses pembuatan *B-Tree* adalah sebagai berikut.

Data	*B-Tree*
7	
2	
1	
4	
9	
5	

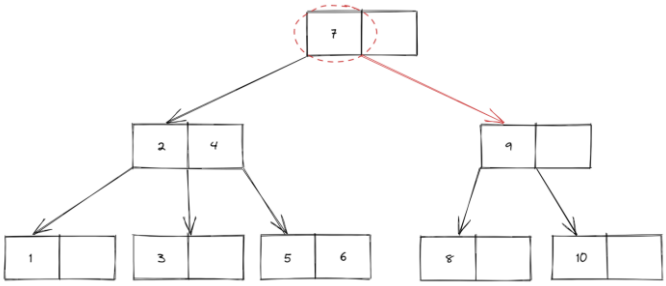
10	
8	
3	
6	

Tabel 1. Ilustrasi pembentukan *B-Tree*
(Sumber: Dokumen pribadi penulis)

Simpul-simpul *B-Tree* pada tabel 1, merupakan simplifikasi dari simpul pada gambar 11, ini semata-mata bertujuan untuk mempermudah visualisasi dan pembacaan, namun fungsionalitasnya tetap sama.

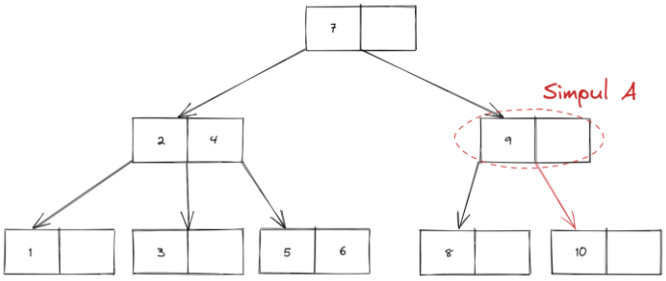
Setelah membentuk *B-Tree* seperti pada tabel 1, misalnya kita ingin melakukan pencarian data 10 pada tabel User, maka proses yang akan terjadi adalah sebagai berikut.

Pertama, dimulai dari akar yang berisi data 7, data target, yaitu 10 > 7 sehingga pencarian akan diteruskan ke simpul anak bagian kanan (gambar 12).



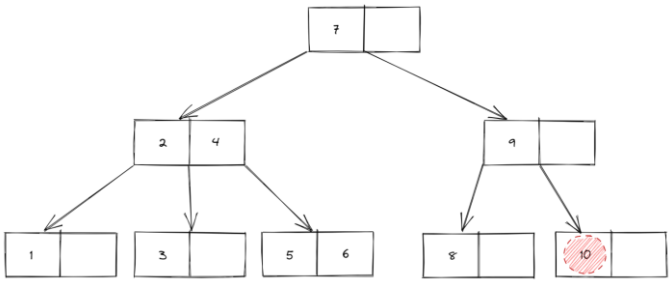
Gambar 12. Langkah pertama pencarian data target
(Sumber: Dokumen pribadi penulis)

Selanjutnya, pada simpul tersebut, sebut saja simpul A, data target, yaitu 10 > 9 sehingga pencarian juga diteruskan ke simpul anak bagian kanan (gambar 13).



Gambar 13. Langkah kedua pencarian data target
(Sumber: Dokumen pribadi penulis)

Ternyata target data 10 berhasil ditemukan pada simpul ini, maka proses pencarian selesai, data diambil dari tabel, dan selanjutnya dikirimkan untuk proses lebih lanjut.



Gambar 14. Data target telah ditemukan
(Sumber: Dokumen pribadi penulis)

Jika kita membandingkan antara banyaknya operasi yang diperlukan untuk memperoleh data 10 pada tabel antara pencarian sekuensial dengan menggunkan *B-Tree*, maka pada pencarian sekuensial kita harus memerlukan 10 kali operasi dengan asumsi data terurut membesar 1,2,3,…,10. Sedangkan, dengan menggunakan *B-Tree* hanya diperlukan 2 kali operasi. Terlihat dengan *B-Tree* secara rata-rata pencarian dapat dilakukan lebih cepat dengan kompleksitas $O(\log N)$, dibandingkan dengan pencarian sekuensial yang memiliki kompleksitas $O(N)$. Walaupun untuk data yang sedikit perbedaannya tidak terlalu jauh, ketika data yang perlu diolah jauh lebih besar seperti kasus sebelumnya, yaitu hingga 1.626.101 data, pencarian secara sekuensial 76x lebih lambat dibandingkan dengan menggunakan *B-Tree* (indeks).

IV. KESIMPULAN

Basis data memiliki peran yang sangat penting dalam proses penyimpanan dan transfer suatu data. Semakin banyaknya data yang disimpan oleh suatu basis data menyebabkan terjadinya penurunan performa yang tentunya harus diatasi. Proses *query* dalam pembacaan suatu data dapat dioptimasi dengan menggunakan indeks. Pembuatan indeks pada suatu kolom akan mempercepat proses *query* data dari kolom tersebut hingga beberapa kali lipat tergantung jumlah dan banyaknya data. *B-Tree* sebagai struktur data yang mendasari indeks menyimpan

key dan pointer-pointer yang mengarah ke data pada memori. Akibat digunakannya *B-Tree* jumlah operasi yang diperlukan untuk mengakses data dari memori berkurang sehingga proses *query* menjadi lebih cepat.

V. Ucapan Terima Kasih

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas bantuan dan rahmat-Nya lah makalah ini dapat diselesaikan. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah menginspirasi dan membantu penulis selama proses penyusunan makalah ini. Terima kasih juga kepada Ibu Dr. Fariska Zakhralativa Ruskanda, S.T., M.T., selaku dosen pengampu mata kuliah IF2120 Matematika Diskrit. Kepada para pembaca makalah ini, terima kasih telah meluangkan waktu untuk membaca, semoga informasi yang terkandung di dalam makalah ini bermanfaat.

Daftar Pustaka

[1] R. Munir, "Pohon (Bagian 1)", *Informatika Rinaldi Munir*, 2022. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf [Diakses: 10 Desember 2022].

[2] R. Munir, "Pohon (Bagian 2)", *Informatika Rinaldi Munir*, 2022. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf [Diakses: 10 Desember 2022].

[3] Admin, "B-Trees", *CS Cornell*. [Online]. Tersedia: https://www.cs.cornell.edu/courses/cs3110/2012sp/recitations/rec25-B-trees/rec25.html [Diakses: 10 Desember 2022].

[4] Z. Mustofa, "Apa Itu Database? Jenis, Fungsi Dan Manfaatnya", *Universitas STEKOM*, 4 Maret 2022. [Online]. Tersedia: http://teknik-informatika-s1.stekom.ac.id/informasi/baca/Apa-itu-Database-Jenis-fungsi-dan-manfaatnya/8de094f78c9551d2eee97e371a249bd714dc83c0 [Diakses: 10 Desember 2022].

[5] Admin, "Pengantar Sistem Basis Data", *Repository DINUS*. [Online]. Tersedia: https://repository.dinus.ac.id/docs/ajar/SBD_1_Sistem_Basis_Data.pdf [Diakses: 10 Desember 2022].

[6] B. Barnhill, "Indexing", *The Data School*, 9 Agustus 2021. [Online]. Tersedia: https://dataschool.com/sql-optimization/how-indexing-works/ [Diakses: 12 Desember 2022].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Desember 2022

I Putu Bakta Hari Sudewa
13521150