

**LAPORAN TUGAS BESAR
MANAJEMEN BASIS DATA**



Nurtias Rahayu (14117086)

**PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA**

2019

DAFTAR ISI

DAFTAR ISI.....	2
LANDASAN TEORI.....	3
1. Tuning : Indexing.....	3
2. Tuning : Setting Configuration DBMS	5
DESKRIPSI PERCOBAAN	9
1. Tuning : Indexing.....	9
HASIL DAN PEMBAHASAN.....	12
1. Tabel Hasil	12
2. Grafik Hasil.....	13
3. Pembahasan Hasil	14
PENUTUP.....	16
1. Kesimpulan	16
2. Saran.....	16

LANDASAN TEORI

Database Tuning adalah sejumlah aktifitas yang dilakukan untuk memperbaiki atau meningkatkan kinerja atau performance sebuah database. Aktifitas tuning ini meliputi banyak aspek dari software hingga hardware, antara lain *I/O Tuning*, *DBMS Tuning*, *Query Tuning*, dan *Database Maintenance*. Masing-masing memiliki tekniknya sendiri-sendiri, dan membutuhkan skill yang mumpuni. Namun kita tetap bisa mempelajari teknik-teknik dasarnya. Dalam artikel ini, kita akan mencoba melakukan Query Tuning dengan bantuan Database Index.

1. Tuning : Indexing

Index adalah sebuah objek dalam sistem database yang dapat mempercepat proses pencarian (query) data. Saat database dibuat tanpa menggunakan index, maka kinerja server database dapat menurun secara drastis. Hal ini dikarenakan resource CPU banyak digunakan untuk pencarian data atau pengaksesan query SQL dengan metode table-scan. Index membuat pencarian data akan lebih cepat dan tidak banyak menghabiskan resource CPU.

Index merupakan objek struktur data tersendiri yang tidak bergantung kepada struktur tabel. Setiap index terdiri dari nilai kolom dan penunjuk (atau ROWID) ke baris yang berisi nilai tersebut. Penunjuk tersebut secara langsung menunjuk ke baris yang tepat pada tabel, sehingga menghindari terjadinya full table-scan. Akan tetapi lebih banyak index pada tabel tidak berarti akan mempercepat query. Semakin banyak index pada suatu tabel menyebabkan kelambatan pemrosesan perintah-perintah DML (Data Manipulation Language), karena setiap terjadi perubahan data maka index juga harus disesuaikan.

Berikut ini adalah beberapa alasan kenapa index diperlukan:

1. Kolom sering digunakan dalam klausa WHERE atau dalam kondisi join
2. Kolom berisi nilai dengan jangkauan yang luas
3. Kolom berisi banyak nilai null
4. Tabel berukuran besar dan sebagian besar query menampilkan data kurang dari 2-4%

Perlu kita perhatikan bahwa terdapat beberapa kondisi dimana tidak diperlukan kehadiran index, yaitu ketika:

1. Table kecil
2. Kolom tidak sering digunakan sebagai kondisi dalam query
3. Kebanyakan query menampilkan data lebih dari 2-4% dari seluruh data
4. Table sering di-update

Menurut Immanuel Chan (2008, p2-11), ada berbagai tipe indexing yang dapat dilakukan, antara lain:

- a. B-Tree Indexes B-Tree Indexes Merupakan teknik indeks yang standar dengan keunggulan untuk primary key dan indeks dengan pemilihan selektif yang tinggi. Indeks dengan teknik B-tree ini dapat digunakan untuk mengembalikan data yang diurutkan berdasarkan indeks pada kolom.
- b. Bitmap indexes Bitmap indexes merupakan teknik yang cocok untuk data dengan kardinalitas yang minimum. Melalui kompresi data, teknik ini dapat menghasilkan row-id dalam jumlah yang besar dengan penggunaan I/O yang minimal. Kombinasi teknik indeks bitmap pada kolom yang tidak diseleksi dapat memberikan efisiensi penggunaan operasi AND dan OR dengan menghasilkan row-id dalam jumlah yang besar dan penggunaan I/O yang minimal. Teknik ini secara khusus efektif dalam query dengan perintah COUNT().

- c. Function-based Indexes Teknik ini dapat membuat akses melalui B-tree pada nilai yang diturunkan dari fungsi yang ada pada data dasar. Teknik ini memiliki batasan dengan penggunaan NULL dan membutuhkan penggunaan optimasi query. Teknik function-based indexes ini secara khusus berguna ketika melakukan query pada kolom-kolom campuran untuk menghasilkan data yang diturunkan atau untuk menanggulangi batasan data yang disimpan dalam basis data.
- d. Partitioned Indexed Indeks dengan partisi dapat dilakukan dengan 2 cara, yakni partisi indeks global dan partisi indeks secara lokal. Indeks global digambarkan dengan hubungan "one-too-many", dengan satu partisi indeks yang akan dipetakan ke banyak partisi tabel. Global indeks hanya dapat digunakan dengan partisi dengan jangkauan tertentu. Indeks lokal digambarkan dengan pemetaan hubungan "one-to-one" antara partisi indeks dan partisi tabel. Secara umum, indeks lokal mengijinkan pendekatan "divide and conquer" untuk menghasilkan eksekusi perintah SQL dengan cepat.

Menghapus index

Index tidak dapat dimodifikasi. Kita harus menghapusnya terlebih dahulu dan menciptakannya kembali. Kita dapat menghapus definisi index dari data dictionary dengan perintah DROP INDEX.

2. Tuning : Setting Configuration DBMS

Penyesuaian basis data terdiri dari sekelompok kegiatan yang digunakan untuk mengoptimalkan dan mengatur kinerja suatu basis data. Ini merujuk pada konfigurasi file database, sistem manajemen basis data (DBMS), serta perangkat keras dan sistem operasi tempat database di-host. Tujuan dari penyetelan basis data adalah untuk memaksimalkan penerapan sumber daya sistem dalam upaya untuk melakukan transaksi seefisien dan secepat mungkin. Sebagian besar DBMS dirancang dengan mempertimbangkan

efisiensi; namun, dimungkinkan untuk meningkatkan kinerja basis data melalui pengaturan dan konfigurasi khusus.

Penyetelan sistem manajemen basis data berpusat di sekitar konfigurasi memori dan sumber daya pemrosesan komputer yang menjalankan DBMS. Ini dapat melibatkan pengaturan interval pemulihan DBMS, menetapkan tingkat kontrol konkurensi, dan menetapkan protokol jaringan mana yang digunakan untuk berkomunikasi di seluruh database. Memori yang digunakan oleh DBMS dialokasikan untuk data, prosedur pelaksanaan, cache prosedur, dan ruang kerja. Karena lebih cepat untuk secara langsung mengakses data dalam memori daripada data pada penyimpanan, dimungkinkan untuk mengurangi waktu akses rata-rata dari transaksi basis data dengan mempertahankan cache data yang berukuran layak. Kinerja basis data juga dapat ditingkatkan dengan menggunakan cache untuk menyimpan prosedur pelaksanaan karena mereka tidak perlu dikompilasi ulang dengan setiap transaksi. Dengan menetapkan sumber daya pemrosesan ke fungsi dan aktivitas tertentu, juga dimungkinkan untuk meningkatkan konkurensi sistem. “Kontrol konkurensi database memastikan bahwa transaksi terjadi dengan cara yang dipesan. Tugas utama dari kontrol ini adalah untuk melindungi transaksi yang dikeluarkan oleh pengguna / aplikasi yang berbeda dari efek satu sama lain. Mereka harus menjaga empat karakteristik transaksi basis data: atomicity, isolasi, konsistensi, dan daya tahan”(About.com).

Penyetelan Input / Output (I / O) adalah komponen utama penyetelan basis data lainnya. Penyetelan I / O terutama berkaitan dengan log transaksi basis data. Log transaksi basis data adalah file yang terkait dengan ruang kerja sementara serta penyimpanan file tabel dan indeks. Log transaksi dan ruang sementara adalah konsumen berat I / O, dan memengaruhi kinerja untuk semua pengguna basis data. Menempatkan mereka dengan tepat sangat penting. Tujuan utama penyetelan database I / O adalah untuk mengoptimalkan dan menyeimbangkan transaksi baca dan tulis sistem

untuk mencapai kecepatan yang meningkat dalam transaksi basis data dan penurunan waktu akses basis data.

Metode lain untuk memastikan bahwa database cepat dan dapat diandalkan adalah Penggunaan RAID dalam pembuatan

RAID 6

Ini menunjukkan tata letak data untuk array RAID-6.

basis data. RAID adalah singkatan dari Redundant Array of Independent Disks. Berikut adalah contoh mengapa RAID lebih unggul dari satu disk. Jika data disimpan pada satu disk, seluruh basis data sepenuhnya bergantung pada satu disk itu; jika gagal, basis data tidak akan ada lagi. Kelemahan lain untuk memilikinya pada satu disk adalah waktu baca / tulis. Satu hard disk hanya bisa sangat cepat. Jika ada banyak data I / O sedang diproses, itu bisa menjadi proses yang panjang. Satu hal yang dilakukan RAID adalah membagi dan mereplikasi data ke beberapa disk independen. Alih-alih memiliki semua telur dalam satu keranjang, kami telah mendiversifikasi risiko atau kegagalan disk kami. Jika kami memiliki array RAID 6 dengan 4 drive, toleransi untuk kegagalan adalah 2 disk. Ini berarti bahwa jika 2 hard drive gagal sepenuhnya, basis data akan tetap berfungsi dengan sempurna. Di atas toleransi kegagalan, sisi positif lain dari penggunaan RAID adalah tugas dilakukan lebih cepat. Ada peningkatan kecepatan yang sama dengan faktor multiplikasi ini: $(n - 2) \times$. Membaca lebih cepat, dan menulis lebih cepat karena alih-alih satu disk mencoba menemukan semua data, tugas dipecah menjadi beberapa bagian, dan setiap hard disk melakukan bagian dari pekerjaan.

Bagian umum lain dari penyetelan basis data berkisar pada pemeliharaan basis data. Pemeliharaan basis data mencakup hal-hal seperti membuat cadangan basis data serta defragmentasi data yang berada di dalam basis data. Ketika sebuah database sedang digunakan, entri log transaksi harus dihapus untuk menciptakan ruang untuk entri masa depan. Karena cadangan

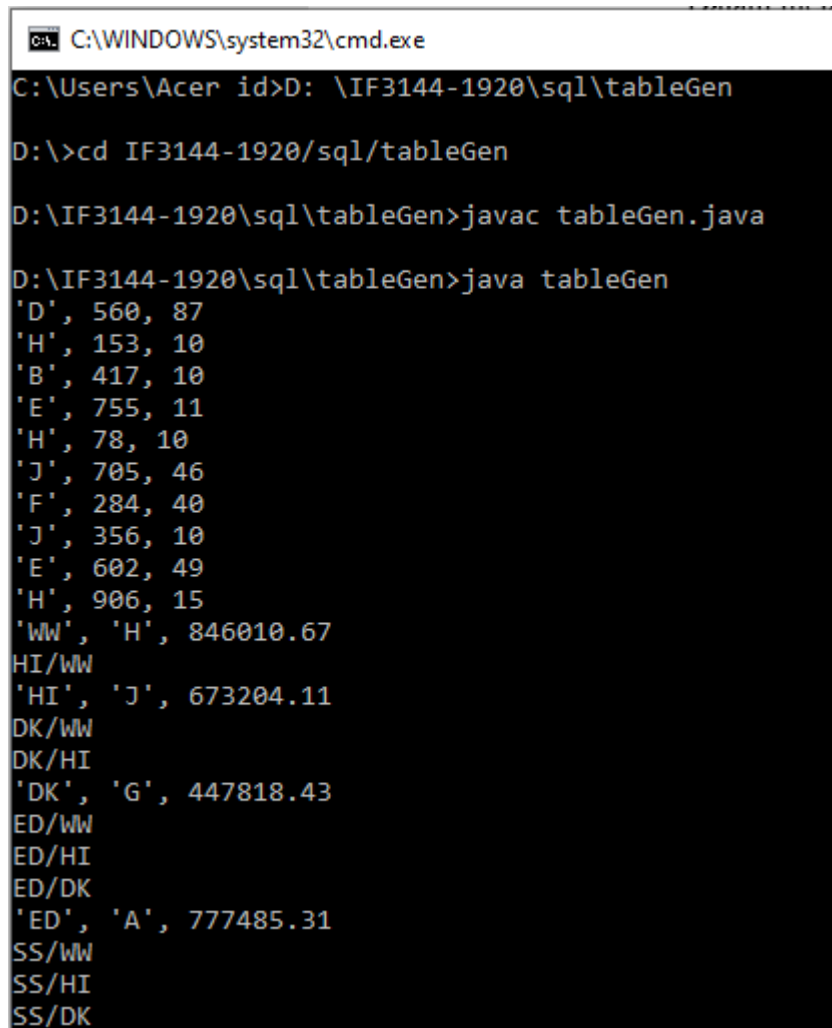
log transaksi reguler berukuran lebih kecil, mereka membutuhkan waktu lebih sedikit untuk menyelesaikan dan, karenanya, mengganggu aktivitas basis data terjadwal untuk periode waktu yang jauh lebih singkat. Seperti halnya sistem komputer, defragmentasi tabel dan indeks basis data sangat meningkatkan efisiensi kemampuan basis data dalam mengakses data. Tingkat fragmentasi basis data tergantung pada sifat data, bagaimana data dimanipulasi, dan jumlah ruang kosong yang tersisa di halaman basis data.

DESKRIPSI PERCOBAAN

1. Tuning : Indexing

Dalam melakukan percobaan tuning index yang dilakukan adalah

- a. Melakukan generate file tableGen.java pada command line untuk mendapatkan SQL yang berisikan data yang akan dimasukkan pada database dan yang nantinya juga akan kita hitung performancenya



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Acer id>D: \IF3144-1920\sql\tableGen
D:\>cd IF3144-1920/sql/tableGen
D:\IF3144-1920\sql\tableGen>javac tableGen.java
D:\IF3144-1920\sql\tableGen>java tableGen
'D', 560, 87
'H', 153, 10
'B', 417, 10
'E', 755, 11
'H', 78, 10
'J', 705, 46
'F', 284, 40
'J', 356, 10
'E', 602, 49
'H', 906, 15
'WW', 'H', 846010.67
HI/WW
'HI', 'J', 673204.11
DK/WW
DK/HI
'DK', 'G', 447818.43
ED/WW
ED/HI
ED/DK
'ED', 'A', 777485.31
SS/WW
SS/HI
SS/DK
```

- b. Setelah itu, kita buat skema database dengan nama yang kita inginkan kemudian import all.sql pada skema yang kita buat.
- c. Setelah itu ketikkan query yang diminta

1. `SELECT * FROM student`
2. `SELECT * FROM student WHERE tot_cred > 30`
3. `SELECT name, dept_name FROM student WHERE
tot_cred > 30`
4. `SELECT * FROM takes JOIN student ON takes.ID =
student.ID JOIN section ON takes.course_id =
section.course_id`
5. `SELECT student.name, student.dept_name, takes.sec_id
AS pengambilan, takes.semester, section.room_number,
section.building, course.course_id, course.dept_name
FROM takes JOIN student ON takes.ID = student.ID JOIN
section ON takes.course_id = section.course_id JOIN
course ON section.course_id = course.course_id;`

- d. Setelah query dijalankan catat waktu yang dibutuhkan query dalam melakukan proses data sebelum melakukan tuning.

C:\WINDOWS\system32\cmd.exe - mysql -u root -p

73025	Josu	ED	16
74250	Kiki	WW	6
75745	Yohan	GT	14
75946	Budi	WW	37
7614	Josu	BN	81
76629	Ahmad	HI	68
77735	Budi	GO	65
78443	Yohan	IF	77
79146	Yohan	FR	106
8057	yuyun	DK	101
8343	Kiki	HI	75
84695	Josu	DK	67
86355	Budi	BN	57
87272	rahmat	WW	10
88108	Ande	GO	1
90176	Budi	HI	92
92508	yuyun	ED	0
93252	Johan	WW	57
93274	rahmat	DK	101
94385	Budi	HI	91
95615	Josu	FR	22
96500	Ande	WW	59
96544	Johan	GO	86
9693	yuyun	DK	121
99619	Kiki	GO	93
99864	Ande	WW	114
99944	yuyun	HI	121

100 rows in set (0.00 sec)

- e. Setelah melakukan semua perhitungan waktu yang dibutuhkan untuk 5 query, selanjutnya buatlah index pada field yang diinginkan sebelum melakukan tuning.

```
MariaDB [mbd]> create index ind_student on student (ID, tot_cred) USING BTREE;
Query OK, 0 rows affected (0.36 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

- f. Ketikkan kembali 5 query yang diminta dan catat waktu prosesnya untuk mengetahui waktu setelah tuning
- g. Lakukan hal diatas untuk data-data lainnya.

HASIL DAN PEMBAHASAN

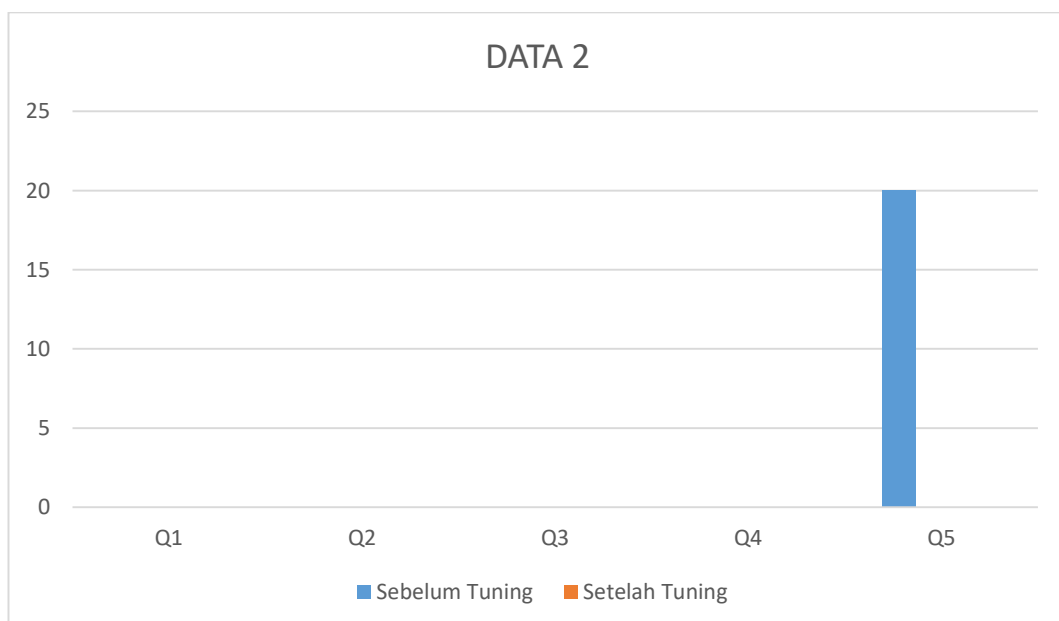
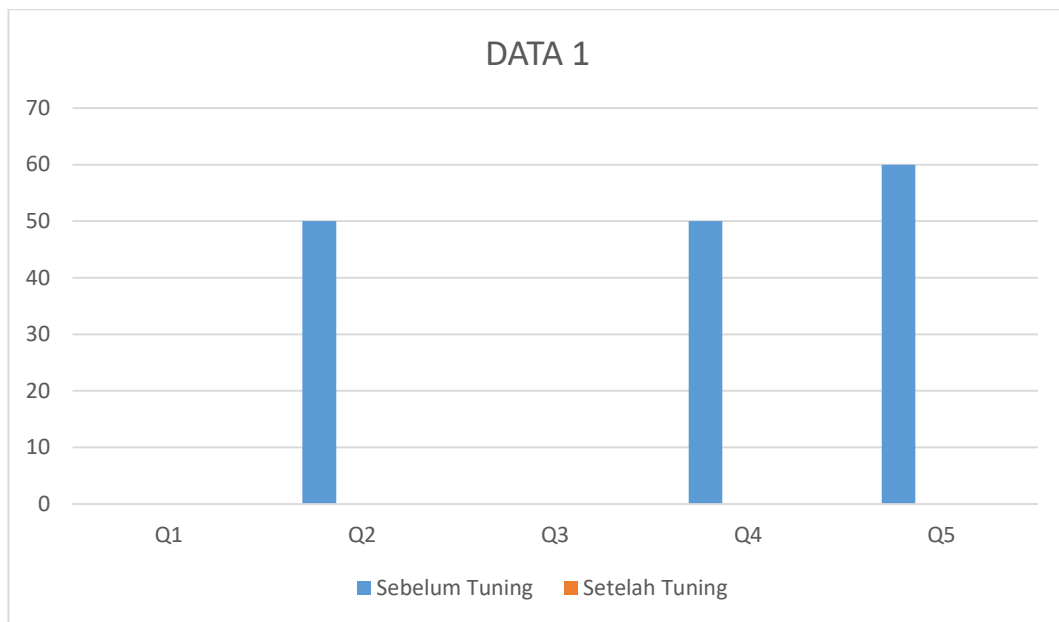
1. Tabel Hasil

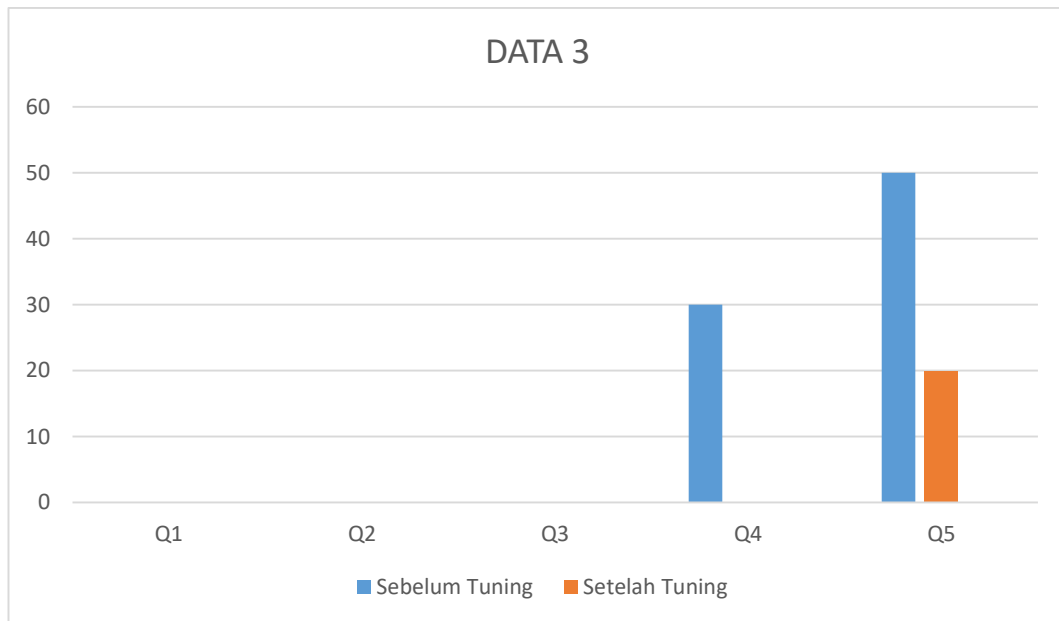
Data	Waktu Sebelum Tuning (ms)					Waktu setelah Tuning (ms)				
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5
1	0	50	0	20	60	0	0	0	0	0
2	0	0	0	0	20	0	0	0	0	0
3	0	0	0	30	50	0	0	0	10	20
4	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-

Keterangan:

1. Data 1: advisor = 100, student = 100, section = 200, takes = 200
2. Data 2: advisor = 200, student = 200, section = 400, takes = 400
3. Data 3: advisor = 500, student = 500, section = 1000, takes = 1000
4. Data 4: advisor = 700, student = 700, section = 20000, takes = 20000
5. Data 5: advisor = 1000, student = 1000, section = 100000, takes = 1000000
6. Data 6: advisor = 1800, student = 1800, section = 180000, takes = 1800000
7. Data 7: advisor = 10000, student = 10000, section = 30000000, takes = 30000000

2. Grafik Hasil





3. Pembahasan Hasil

Database Tuning adalah sejumlah aktifitas yang dilakukan untuk memperbaiki atau meningkatkan kinerja atau performance sebuah database. Aktifitas tuning ini meliputi banyak aspek dari software hingga hardware, antara lain I/O Tuning, DBMS Tuning, Query Tuning, dan Database Maintenance. Tuning index merupakan salah satu teknik dari DBMS Tuning.

Pada percobaan diatas menggunakan teknik indexing untuk tuning, dimana saya membuat index dengan menggunakan teknik B-Tree pada salah satu atribut pada tabel yang sering diakses dalam query. Setelah melakukan indexing dapat dilihat pada tabel dan grafik ternyata ada perubahan waktu eksekusi dengan sebelum indexing, dimana saat sebelum melakukan tuning index waktu yang dibutuhkan dalam mengeksekusi query rata-rata relative lebih lama dibanding setelah melakukan indexing. Karena dengan indexing DBMS akan lebih mudah untuk mencari dan mengakses data-data yang ada.

Dikarenakan perangkat yang tidak mendukung, untuk data 4-7 tidak dapat dilakukan setelah beberapa kali dicoba tetap tidak bisa dilakukan.

PENUTUP

1. Kesimpulan

Adapun kesimpulan yang saya dapatkan setelah melakukan percobaan adalah sebagai berikut: a) Database tuning sangatlah diperlukan untuk meningkatkan performansi DBMS, terutama DBMS yang memiliki database yang banyak. b) Tuning indexing merupakan salah satu teknik yang dapat dilakukan untuk melakukan database tuning. c) Dibutuhkan perangkat yang mendukung untuk mengolah data yang besar.

2. Saran

Adapun saran yang dapat saya berikan setelah melakukan percobaan adalah sebaiknya gunakan perangkat yang memiliki spesifikasi yang tinggi dan mendukung jika ingin melakukan pengolahan database dengan jumlah data yang besar.