

**LAPORAN TUGAS BESAR**  
**DATABASE TUNING**  
**MANAJEMEN BASIS DATA - RA**



**Disusun oleh :**  
**Nurul Fauzia Azizah (14117071)**

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**INSTITUT TEKNOLOGI SUMATERA**

**2019**

# Daftar Isi

Daftar Isi .....	ii
BAB I: PENDAHULUAN.....	3
1. Teori Dasar .....	3
1.1. Tunning: Indexing.....	3
1.2. Tunning: Setting Configuration DBMS .....	3
2. Deskripsi Percobaan .....	3
2.1. Sebelum Tunning.....	3
2.2. Tunning: Indexing.....	5
2.3. Tunning: Setting Configuration DBMS .....	7
BAB II DESKRIPSI PERCOBAAN .....	6
1. Tuning.....	8
BAB III: HASIL DAN PEMBAHASAN.....	12
3.1 Hasil dan Pembahasan.....	12
Daftar Pustaka.....	10

# BAB I PENDAHULUAN

## 1. Dasar Teori

### 1.1. Tuning: Indexing

Database tuning adalah aktivitas untuk membuat aplikasi database berjalan dengan lebih cepat. Lebih cepat artinya nilai throughput lebih besar walaupun response time-nya lebih rendah. Index adalah struktur data yang mendukung akses efisien ke data. Index tuning berarti membuat index yang tepat untuk mempercepat query atau updates. Index tuning melihat ke query dan updates lama lalu merekomendasikan index mana yang paling baik untuk beban kerja.

#### Kerugian Indeks

Indeks adalah hambatan kinerja ketika tiba saatnya untuk mengubah catatan. Setiap kali query memodifikasi data dalam tabel, indeks pada data juga harus berubah. Untuk mencapai jumlah indeks yang tepat akan membutuhkan pengujian dan pemantauan database Anda untuk melihat di mana keseimbangan terbaik berada. Sistem statis, di mana basis data banyak digunakan untuk pelaporan, dapat membeli lebih banyak indeks untuk mendukung query hanya baca. Basis data dengan jumlah transaksi yang banyak untuk mengubah data akan membutuhkan lebih sedikit indeks untuk memungkinkan throughput yang lebih tinggi. Indeks juga menggunakan ruang disk. Ukuran pasti akan tergantung pada jumlah catatan dalam tabel serta jumlah dan ukuran kolom dalam indeks. Umumnya ini bukan masalah utama karena ruang disk mudah ditukar untuk kinerja yang lebih baik. Membangun Indeks Terbaik Ada sejumlah pedoman untuk membangun indeks yang paling      Anda. Dari kolom yang Anda pilih ke nilai data di dalamnya, pertimbangkan hal-hal berikut saat memilih indeks untuk tabel Anda.

#### Short Key

Memiliki indeks pendek bermanfaat karena dua alasan. Pertama, pekerjaan basis data bersifat intensif disk. Kunci indeks yang lebih besar akan menyebabkan database melakukan lebih banyak pembacaan disk, yang membatasi throughput. Kedua, karena entri indeks sering terlibat dalam perbandingan, entri yang lebih kecil lebih mudah untuk dibandingkan. Kolom integer tunggal membuat kunci indeks terbaik mutlak

karena integer kecil dan mudah untuk membandingkan database. String karakter, di sisi lain, membutuhkan perbandingan karakter dengan karakter dan perhatian pada pengaturan susunan.

### Distinct Keys

Indeks yang paling efektif adalah indeks dengan persentase kecil dari nilai yang diduplikasi. Sebagai analogi, pikirkan sebuah buku telepon untuk sebuah kota di mana hampir setiap orang memiliki nama belakang Smith. Buku telepon di kota ini tidak terlalu berguna jika disortir berdasarkan nama belakang, karena Anda hanya dapat mendiskon sejumlah kecil catatan ketika Anda mencari Smith.

Indeks dengan persentase tinggi dari nilai unik adalah indeks selektif. Jelas, indeks unik sangat selektif karena tidak ada entri duplikat. Banyak basis data akan melacak statistik tentang setiap indeks sehingga mereka tahu seberapa selektif setiap indeks. Basis data menggunakan statistik ini saat membuat rencana eksekusi untuk query.

### Covering Queries

Indeks umumnya hanya berisi nilai data untuk kolom yang diindeks dan penunjuk kembali ke baris dengan data lainnya. Ini mirip dengan indeks dalam sebuah buku: indeks hanya berisi kata kunci dan kemudian referensi halaman Anda dapat beralih ke untuk sisa informasi. Umumnya database harus mengikuti petunjuk dari indeks kembali ke baris untuk mengumpulkan semua informasi yang diperlukan untuk permintaan. Namun, jika indeks berisi semua kolom yang diperlukan untuk kueri, database dapat menyimpan disk yang dibaca dengan tidak kembali ke tabel untuk informasi lebih lanjut.

### Clustered Indexes

Banyak basis data memiliki satu indeks khusus per tabel di mana semua data dari satu baris ada dalam indeks. SQL Server menyebut indeks ini sebagai indeks berkerumun. Alih-alih indeks di bagian belakang buku, indeks berkerumun lebih mirip dengan buku telepon karena setiap entri indeks berisi semua informasi yang Anda butuhkan, tidak ada referensi untuk mengikuti untuk mengambil nilai data tambahan. Sebagai aturan umum, setiap tabel non-sepele harus memiliki indeks berkerumun. Jika Anda hanya membuat satu indeks untuk tabel, buat indeks sebagai indeks

berkerumun. Dalam SQL Server, membuat kunci utama akan secara otomatis membuat indeks berkerumun (jika tidak ada) menggunakan kolom kunci utama sebagai kunci indeks. Indeks Clustered adalah indeks yang paling efektif (ketika digunakan, mereka selalu mencakup permintaan), dan dalam banyak sistem database akan membantu database mengelola ruang yang dibutuhkan untuk menyimpan tabel secara efisien.

Saat memilih kolom atau kolom untuk indeks berkerumun, berhati-hatilah untuk memilih kolom dengan data statis. Jika Anda memodifikasi catatan dan mengubah nilai kolom dalam indeks berkerumun, database mungkin perlu memindahkan entri indeks (untuk menjaga entri dalam urutan diurutkan). Ingat, entri indeks untuk indeks berkerumun berisi semua nilai kolom, sehingga memindahkan entri sebanding dengan mengeksekusi pernyataan DELETE diikuti oleh INSERT, yang jelas dapat menyebabkan masalah kinerja jika sering dilakukan. Untuk alasan ini, indeks berkerumun sering ditemukan pada kolom kunci utama atau asing. Nilai kunci jarang, jika pernah, berubah.

## 1.2. Tuning: Setting Configuration DBMS

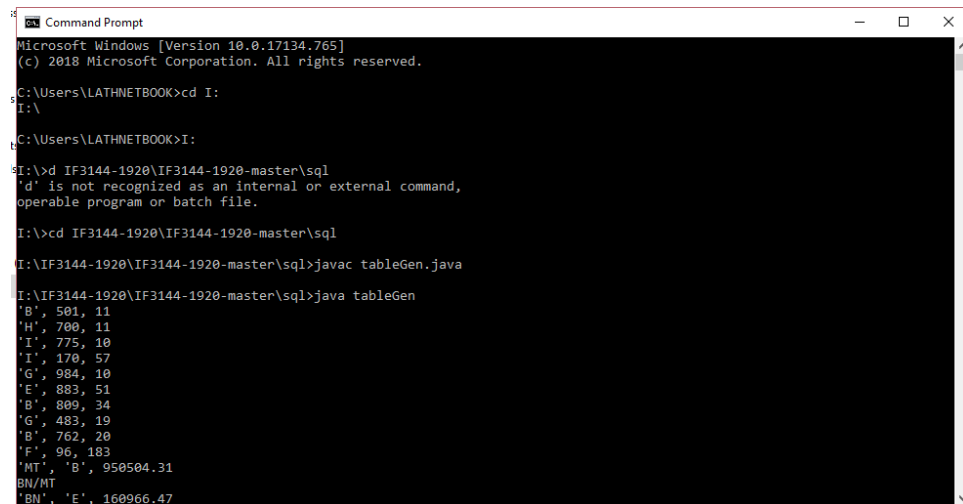
Database tuning menjelaskan sekelompok kegiatan yang digunakan untuk mengoptimalkan dan menyeragamkan kinerja suatu basis data. Biasanya tumpang tindih dengan penyetelan kueri, tetapi merujuk pada desain file basis data, pemilihan aplikasi sistem manajemen basis data (DBMS), dan konfigurasi lingkungan basis data (sistem operasi, CPU, dll.). Penyesuaian basis data bertujuan untuk memaksimalkan penggunaan sumber daya sistem untuk melakukan pekerjaan seefisien dan secepat mungkin. Sebagian besar sistem dirancang untuk mengelola penggunaan sumber daya sistem mereka, tetapi masih ada banyak ruang untuk meningkatkan efisiensinya dengan menyesuaikan pengaturan dan konfigurasi mereka untuk database dan DBMS.

## BAB II DESKRIPSI PERCOBAAN

### 3.1 Tuning

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



```
Microsoft Windows [Version 10.0.17134.765]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\LATHNETBOOK>cd I:
I:\

C:\Users\LATHNETBOOK>I:

I:\>cd IF3144-1920\IF3144-1920-master\sql
I:\IF3144-1920\IF3144-1920-master\sql>javac tableGen.java
I:\IF3144-1920\IF3144-1920-master\sql>java tableGen
'B', 501, 11
'H', 700, 11
'I', 775, 10
'I', 170, 57
'G', 984, 10
'E', 883, 51
'B', 809, 34
'G', 483, 19
'B', 762, 20
'E', 96, 183
'HT', 'B', 950504.31
BN/HT
'BN', 'E', 160966.47
```

Gambar 4 compile java tableGen pada Data 1

- 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
Database changed
MariaDB [datambd1]> select*from student;
```

ID	name	dept_name	tot_cred
10900	Adri	SS	101
12265	Kiki	SS	83
12352	yuyun	RR	63
13147	Kiki	DE	107
14994	Josu	DE	89
15251	Johan	DE	31
15833	Ande	HI	56
1697	rahmat	RR	112
17000	Ande	GF	71
18011	Budi	HI	12
18214	Kiki	RR	65
18866	Budi	HI	127
19263	Josu	MT	72
19603	Budi	BN	18
2181	Ahmad	WW	32
21871	Kiki	WW	3
21943	Ahmad	RR	110
22244	Johan	SS	66
22929	Josu	GO	105
23131	rahmat	BN	44
25632	Ande	DE	83
27075	Johan	HI	30
28503	Ande	GO	100
28655	Adri	MT	17
28806	Ahmad	SS	66
29230	Yohan	RR	124
29553	Ande	DE	123
2991	Budi	IF	13
29950	Johan	GO	33
30293	Adri	DE	105
30519	Ande	HI	25
32413	rahmat	MT	8
32762	Ande	WW	45
35309	Ande	GF	125
35857	Ahmad	HI	14
37259	Ahmad	HI	16
37959	Adri	GO	95
38820	rahmat	BN	80
39485	Budi	GO	63

Gambar 4 running query pada Data 1

- e. Untuk mencatatn waktu dapat kita lihat dengan menggunakan perintah ‘set spiling = 1;’ dan setelah runing dengan perintah ‘show profiles’

```

C:\WINDOWS\system32\cmd.exe - mysql -u root -p
7383 | Johan | DE | 18 |
74007 | rahmat | RR | 30 |
75276 | Kiki | MT | 117 |
75335 | Yohan | HI | 79 |
75459 | rahmat | GO | 110 |
75732 | Budi | RR | 119 |
75798 | rahmat | HI | 25 |
76577 | Budi | MT | 91 |
77380 | Budi | RR | 71 |
77498 | Ahmad | HI | 124 |
78556 | Adri | MT | 111 |
78900 | Yohan | MT | 29 |
78968 | Kiki | IF | 111 |
80895 | Ahmad | GO | 59 |
81274 | yuyun | HI | 123 |
81711 | Budi | RR | 79 |
81733 | rahmat | MT | 76 |
81968 | Johan | DE | 108 |
83351 | Josu | SS | 129 |
84540 | Adri | GO | 118 |
8557 | Yohan | WM | 115 |
88292 | Kiki | BN | 92 |
88732 | yuyun | SS | 20 |
9053 | Josu | GF | 31 |
91595 | Johan | BN | 56 |
92135 | Ahmad | MT | 5 |
95341 | Ahmad | MT | 120 |
95817 | Josu | DE | 90 |
96164 | Ahmad | RR | 17 |
96619 | Kiki | HI | 116 |
975 | Kiki | HI | 78 |
9779 | Ahmad | SS | 42 |
99257 | Adri | HI | 110 |
99332 | Ahmad | HI | 71 |
-----+-----+
100 rows in set (0.00 sec)

MariaDB [datambd1]> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.00085755 | select*from student |
+-----+-----+-----+
1 row in set (0.00 sec)

```

Gambar 4 Pehitungan sebelum tuning pada Data 1

f. Lakukan hal diatas untuk data-data lainnya.

```

Command Prompt - mysql -u root -p
MariaDB [datambd1]> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.00101891 | SELECT * FROM student |
+-----+-----+-----+
| 2 | 0.04860793 | SELECT * FROM student WHERE tot_cred > 30 |
+-----+-----+-----+
| 3 | 0.00092711 | SELECT 'name', dept_name FROM student WHERE tot_cred > 30 |
+-----+-----+-----+
| 4 | 0.09009943 | SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = s |
| section.course_id |
+-----+-----+-----+
| 5 | 0.07772137 | SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = s |
| section.course_id |
+-----+-----+-----+
| 6 | 0.03732804 | SELECT student.`name`,student.dept_name,takes.sec_id AS pengambilan,takes.semester,section.roo |
| m_number,section.building,course.course_id,course.dept_name FROM takes JOIN student ON takes.ID = student.ID JOIN sectio |
| n ON takes.course_id = section.course_id JOIN course ON section.course_id = course.cou |
+-----+-----+-----+
| 7 | 0.00071102 | SET GLOBAL query_chace_size = 268435456 |
+-----+-----+-----+

```

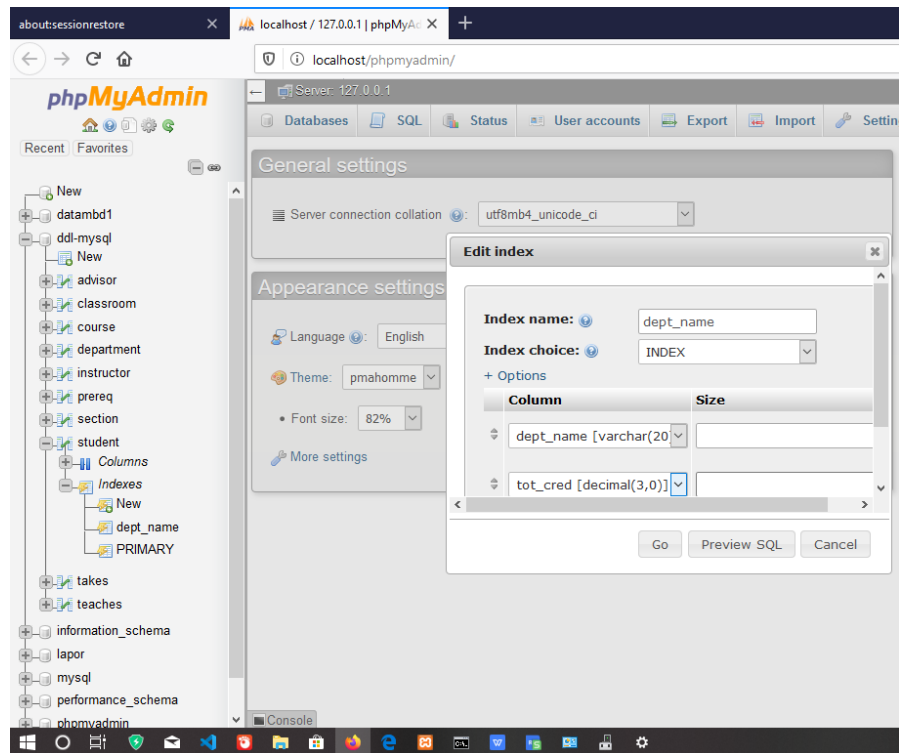
Gambar 4 Pehitungan sebelum dituning pada Data 1

### 1.3. Tunning: Indexing

Dalam tuning indexing, langkah pertama yang harus dilakukan adalah membuat index pada field yang dipilih.

a. Dengan menambahkan colom indeks pada database student



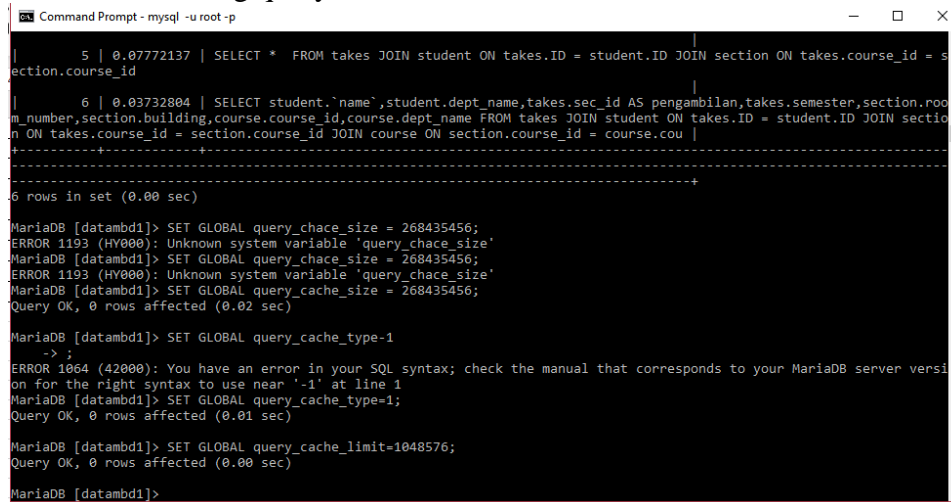


Gambar 4 Penambahan Index pada Data 1

- g. selanjutnya buatlah index pada field yang diinginkan sebelum melakukan tuning.
- h. Ketikkan kembali 5 query yang diminta dan catat waktu prosesnya untuk mengetahui waktu setelah tunin.

## 1.4. Tuning: Setting Configuration DBMS

### a. Melakukan setting query



```
Command Prompt - mysql -u root -p

5 | 0.07772137 | SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = s
ection.course_id

6 | 0.03732804 | SELECT student.`name`, student.dept_name, takes.sec_id AS pengambilan, takes.semester, section.roo
m_number, section.building, course.course_id, course.dept_name FROM takes JOIN student ON takes.ID = student.ID JOIN sectio
n ON takes.course_id = section.course_id JOIN course ON section.course_id = course.cou

-----+
6 rows in set (0.00 sec)

MariaDB [datambd1]> SET GLOBAL query_cache_size = 268435456;
ERROR 1193 (HY000): Unknown system variable 'query_cache_size'
MariaDB [datambd1]> SET GLOBAL query_cache_size = 268435456;
ERROR 1193 (HY000): Unknown system variable 'query_cache_size'
MariaDB [datambd1]> SET GLOBAL query_cache_size = 268435456;
Query OK, 0 rows affected (0.02 sec)

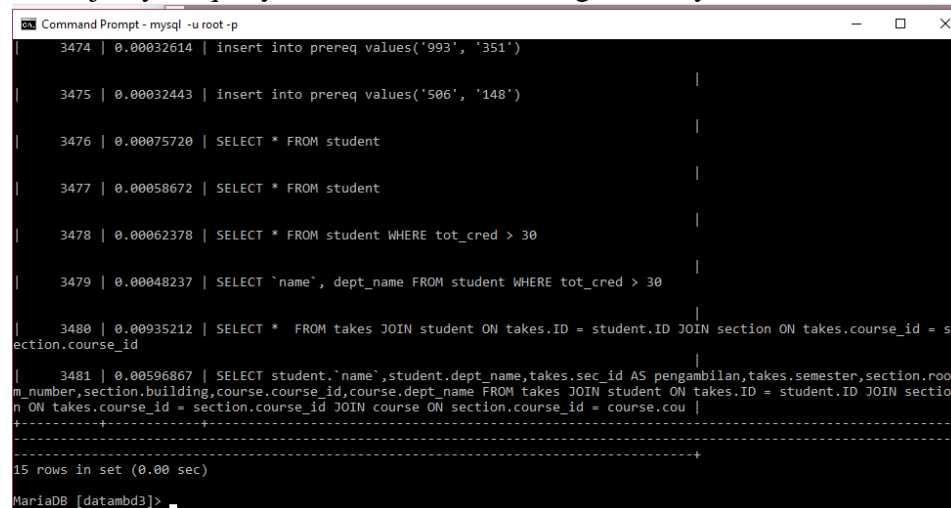
MariaDB [datambd1]> SET GLOBAL query_cache_type=1
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server versi
on for the right syntax to use near '-1' at line 1
MariaDB [datambd1]> SET GLOBAL query_cache_type=1;
Query OK, 0 rows affected (0.01 sec)

MariaDB [datambd1]> SET GLOBAL query_cache_limit=1048576;
Query OK, 0 rows affected (0.00 sec)

MariaDB [datambd1]>
```

Gambar 4 Penambahan tuning DBMS pada Data 1

### b. Selanjutnya 5 query dimasukan dan dihitung waktunya.



```
Command Prompt - mysql -u root -p

3474 | 0.00032614 | insert into prereq values('993', '351')

3475 | 0.00032443 | insert into prereq values('506', '148')

3476 | 0.00075720 | SELECT * FROM student

3477 | 0.00058672 | SELECT * FROM student

3478 | 0.00062378 | SELECT * FROM student WHERE tot_cred > 30

3479 | 0.00048237 | SELECT `name`, dept_name FROM student WHERE tot_cred > 30

3480 | 0.00935212 | SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = s
ection.course_id

3481 | 0.00596867 | SELECT student.`name`, student.dept_name, takes.sec_id AS pengambilan, takes.semester, section.roo
m_number, section.building, course.course_id, course.dept_name FROM takes JOIN student ON takes.ID = student.ID JOIN sectio
n ON takes.course_id = section.course_id JOIN course ON section.course_id = course.cou

-----+
15 rows in set (0.00 sec)

MariaDB [datambd3]>
```

Gambar 4 Pehitungan tuning DBMS pada data1

## BAB III Hasil dan Pembahasan

### 3.1. Hasil dan Pembahasan

Waktu Sebelum Tunning (ms)					Waktu Sesudah Tuning index (ms)					Waktu Sesudah Tuning DBMS(ms)				
Query 1	Query 2	Query 3	Query 4	Query 5	Query 1	Query 2	Query 3	Query 4	Query 5	Query 1	Query 2	Query 3	Query 4	Query 5
0.027316	0.001607	0.001612	0.004211	0.003209	0.001019	0.048608	0.000927	0.077721	0.037328	0.000817	0.000843	0.000811	0.003678	0.00297
0.000387	0.002681	0.000917	0.01025	0.006547	0.001081	0.001202	0.001255	0.010612	0.010026	0.001083	0.001012	0.001022	0.009192	0.007194
0.000555	0.003732	0.002077	0.039988	0.062588	0.000757	0.000624	0.000624	0.009352	0.005969	0.004672	0.001801	0.002693	0.059225	0.021763

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. Dari hasil tuning dengan indexing pada data-data di atas, didapatkan bahwa rata-rata data mengalami kenaikan tingkat kecepatan dari sebelum dan sesudah indexing. Hal ini sesuai dengan studi literatur yaitu dengan melakukan tuning index dapat meningkatkan kecepatan query.

## Daftar Pustaka

Silberzchatz, Database System Concept, 6<sup>th</sup> Edition