

LAPORAN TUGAS BESAR
TUNING DATABASE



Dosen Pengampu :

Achmad Luky Ramdani, S.Kom.M.Kom.

Disusun Oleh :

Rizqun Rizal Ahsani (14117133)

Kelas RB

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA

2019

DAFTAR ISI

DAFTAR ISI.....	ii
BAB I.....	3
STUDI LITERATUR.....	3
1.1 Tuning: Indexing.....	3
1.2 Tuning: Setting Configuration DBMS.....	3
BAB II	4
DESKRIPSI PERCOBAAN	4
2.1 Tuning: Indexing.....	4
BAB III	15
HASIL DAN PEMBAHASAN	15
3.1 Tabel Hasil	15
3.2 Grafik Hasil.....	15
3.3 Pembahasan Hasil	17
DAFTAR PUSTAKA	18

BAB I

STUDI LITERATUR

1.1 Tuning: Indexing

Tuning basis data adalah suatu aktivitas yang dilakukan untuk memberikan peningkatan kinerja pada basis data untuk meningkatkan *throughput*, menurunkan *response time*, dan memiliki *availability* yang tinggi. Dalam melakukan tuning, terdapat banyak hal yang mempengaruhi performa, antara lain adalah dari sisi *hardware*, pengaturan sistem operasi, pengaturan *database server parameter*, *database design*, *indexing* pada tabel *database*, dan *sql statement*. Dengan melakukan tuning terhadap faktor-faktor tersebut, maka performa dari basis data dapat ditingkatkan.

Indexing adalah salah satu cara yang dapat dilakukan dalam meningkatkan performa dari basis data. Indexing adalah sebuah data struktur yang menyimpan nilai spesifik sebuah kolom pada sebuah tabel untuk mempercepat proses pencarian data. Teknik-teknik dalam melakukan indexing ada banyak jenisnya, antara lain adalah Hash indexes, B-tree indexes, dan Bitmap indexes.

1.2 Tuning: Setting Configuration DBMS

Sistem tuning terjadi pada tingkat tertinggi dan memiliki dampak terbesar pada kesehatan secara keseluruhan aplikasi database, karena setiap aplikasi tergantung pada sistem. Kita akan mendefinisikan sistem terdiri dari DBMS itu sendiri dan semua komponen terkait yang dipercayakan. DBMS dapat dan harus bisa disetel untuk menjamin kinerja yang optimal. Cara di mana perangkat lunak DBMS terinstal, memori, disk, CPU, sumber daya lainnya, dan opsi konfigurasi dapat mempengaruhi kinerja aplikasi database. Dan tentunya DBMS harus bisa di seting agar bisa menjamin performa sistem tetap optimal.

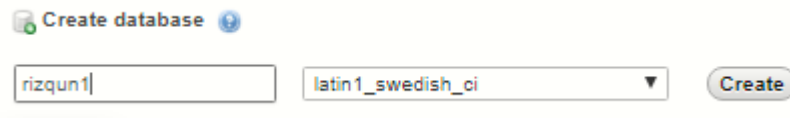
BAB II

DESKRIPSI PERCOBAAN

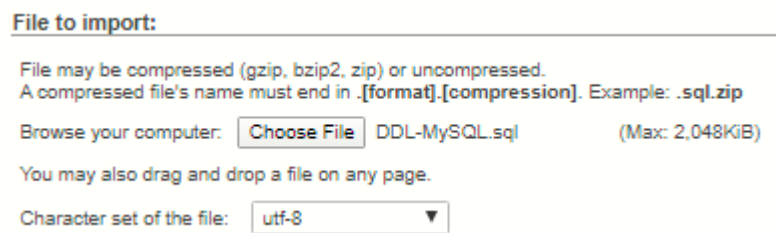
2.1 Tuning: Indexing

A. Data 1

1. Buat database dengan nama rizqun1 untuk pengerjaan data pertama



2. Import DDL-MySQL.sql untuk menambahkan tabel-tabel pada database.



3. Periksa pada tableGen.java, apakah jumlah data yang akan di load sudah sesuai dengan yang diminta.

```
public static void main(String[] args) {  
    int classroom = 10;  
    int department = 10;  
    int course = 200;  
    int instructor = 50;  
    int teaches = 100;  
    int advisor = 100;  
    int student = 100;  
    int section = 200;  
    int takes = 200;  
    int prereq = 100;  
    int timeSlot = 10;  
    int i = 0, j = 0, r = 0, c = 0, x = 0, y = 0;  
    boolean tryValue = true;  
    String b = "", d = "", s = "";  
    fillArrays();  
}
```

4. Jika sudah sesuai, compile tableGen.java dengan menggunakan command prompt.

```
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>javac tableGen.java  
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>java tableGen
```

5. Maka akan terbentuk file all.sql, kemudian import all.sql tersebut kedalam database.

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in `.[format].[compression]`. Example: `.sql.zip`

Browse your computer: all.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

6. Jalankan query 1 (sebelum tuning) dan catat waktunya.

✓ Showing rows 0 - 24 (100 total, Query took 0.0030 seconds.)

```
SELECT * FROM student
```

Didapatkan waktu sebesar 0.0030 detik.

7. Jalankan query 2 (sebelum tuning) dan catat waktunya.

✓ Showing rows 0 - 24 (67 total, Query took 0.0040 seconds.)

```
SELECT * FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0040 detik.

8. Jalankan query 3 (sebelum tuning) dan catat waktunya.

✓ Showing rows 0 - 24 (67 total, Query took 0.0041 seconds.)

```
SELECT dept_name FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0041 detik.

9. Jalankan query 4 (sebelum tuning) dan catat waktunya.

✓ Showing rows 0 - 24 (373 total, Query took 0.0070 seconds.)

```
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id
```

Didapatkan waktu sebesar 0.0070 detik.

10. Jalankan query 5 (sebelum tuning) dan catat waktunya.

✓ Showing rows 0 - 24 (373 total, Query took 0.0114 seconds.)

```
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
```

Didapatkan waktu sebesar 0.0114 detik.

11. Lakukan tuning indexing dengan memasukkan query seperti dibawah.

```
1 CREATE INDEX indexing ON takes(course_id) USING BTREE
```



```
1 CREATE INDEX indexing ON section(course_id) USING BTREE
```

12. Jalankan query 1 (setelah tuning) dan catat waktunya.

```
Showing rows 0 - 24 (100 total, Query took 0.0029 seconds.)
select * from student
```

Didapatkan waktu sebesar 0.0029 detik.

13. Jalankan query 2 (setelah tuning) dan catat waktunya.

```
Showing rows 0 - 24 (67 total, Query took 0.0039 seconds.)
SELECT * FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0039 detik.

14. Jalankan query 3 (setelah tuning) dan catat waktunya.

```
Showing rows 0 - 24 (67 total, Query took 0.0037 seconds.)
SELECT dept_name FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0037 detik.

15. Jalankan query 4 (setelah tuning) dan catat waktunya.

```
Showing rows 0 - 24 (373 total, Query took 0.0063 seconds.)
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id
```

Didapatkan waktu sebesar 0.0063 detik.


16. Jalankan query 5 (setelah tuning) dan catat waktunya.

```
Showing rows 0 - 24 (373 total, Query took 0.0112 seconds.)
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
```

Didapatkan waktu sebesar 0.0112 detik.

B. Data 2

1. Buat database dengan nama rizqun2 untuk pengerjaan data kedua

Create database 

2. Import DDL-MySQL.sql untuk menambahkan tabel-tabel pada database.

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.

A compressed file's name must end in `.[format].[compression]`. Example: `.sql.zip`

Browse your computer: DDL-MySQL.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

- Ubah tableGen.java dan sesuaikan dengan jumlah data yang diminta.

```
public static void main(String[] args) {  
    int classroom = 10;  
    int department = 10;  
    int course = 200;  
    int instructor = 50;  
    int teaches = 100;  
    int advisor = 200;  
    int student = 200;  
    int section = 400;  
    int takes = 400;  
    int prereq = 100;  
    int timeSlot = 10;  
    int i = 0, j = 0, r = 0, c = 0, x = 0, y = 0;  
    boolean tryValue = true;  
    String b = "", d = "", s = "";  
    fillArrays();  
}
```

- Jika sudah sesuai, compile tableGen.java dengan menggunakan command prompt.

```
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>javac tableGen.java  
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>java tableGen
```

- Maka akan terbentuk file all.sql, kemudian import all.sql tersebut kedalam database.

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in .[format].[compression]. Example: .sql.zip

Browse your computer: all.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

- Jalankan query 1 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (200 total, Query took 0.0031 seconds.)  
SELECT * FROM "student"
```

Didapatkan waktu sebesar 0.0031 detik.

- Jalankan query 2 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (155 total, Query took 0.0040 seconds.)  
SELECT * FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0040 detik.

- Jalankan query 3 (sebelum tuning) dan catat waktunya.

Showing rows 0 - 24 (155 total, Query took 0.0040 seconds.)

```
SELECT dept_name FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0040 detik.

9. Jalankan query 4 (sebelum tuning) dan catat waktunya.

Showing rows 0 - 24 (1154 total, Query took 0.0068 seconds.)

```
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id
```

Didapatkan waktu sebesar 0.0068 detik.

10. Jalankan query 5 (sebelum tuning) dan catat waktunya.

Showing rows 0 - 24 (1154 total, Query took 0.0111 seconds)

```

SELECT student_name, student_dept_name, takes.sei_id AS poignante, takes.semester, section.row_number, section.building, course.course_id, course.dept_name FROM takes JOIN student ON takes.student_id = student.id JOIN section ON takes.course_id = section.course_id JOIN course ON takes.course_id = section.course_id

```

Didapatkan waktu sebesar 0.0111 detik.

11. Lakukan tuning indexing dengan memasukkan query seperti dibawah.

```
1 CREATE INDEX indexing ON takes(course_id) USING BTREE
```

```
1 CREATE INDEX indexing ON section(course_id) USING BTREE
```

12. Jalankan query 1 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (200 total, Query took 0.0030 seconds.)

```
SELECT * FROM 'student' WHERE 1
```

Didapatkan waktu sebesar 0.0030 detik.

13. Jalankan query 2 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (155 total, Query took 0.0035 seconds.)

```
SELECT * FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0035 detik.

14. Jalankan query 3 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (155 total, Query took 0.0035 seconds.)

```
SELECT dept_name FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0035 detik.

15. Jalankan query 4 (setelah tuning) dan catat waktunya.


```

Showing rows 0 - 24 (1154 total, Query took 0.0062 seconds.)
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id

```

Didapatkan waktu sebesar 0.0062 detik.

16. Jalankan query 5 (setelah tuning) dan catat waktunya.

```

Showing rows 0 - 24 (1154 total, Query took 0.0106 seconds.)
SELECT student_name, student_dept_name, takes_seq_id AS pengambil, 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 WHERE takes.course_id = 200

```

Didapatkan waktu sebesar 0.0106 detik.

C. Data 3

1. Buat database dengan nama rizqun3 untuk pengerjaan data ketiga

Create database

rizqun3 latin1_swedish_ci

Create

2. Import DDL-MySQL.sql untuk menambahkan tabel-tabel pada database.

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in `[format].[compression]`. Example: `.sql.zip`

Browse your computer: DDL-MySQL.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file: utf-8

3. Ubah tableGen.java dan sesuaikan dengan jumlah data yang diminta.

```

public static void main(String[] args) {
    int classroom = 10;
    int department = 10;
    int course = 200;
    int instructor = 50;
    int teaches = 100;
    int advisor = 500;
    int student = 500;
    int section = 1000;
    int takes = 1000;
    int prereq = 100;
    int timeSlot = 10;
    int i = 0, j = 0, r = 0, c = 0, x = 0, y = 0;
    boolean tryValue = true;
    String b = "", d = "", s = "";
    fillArrays();
}

```

4. Jika sudah sesuai, compile tableGen.java dengan menggunakan command prompt.

```
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>javac tableGen.java
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>java tableGen
```

5. Maka akan terbentuk file all.sql, kemudian import all.sql tersebut kedalam database.

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in .[format].[compression]. Example: .sql.zip

Browse your computer: all.sql (Max: 2,048KB)

You may also drag and drop a file on any page.

Character set of the file:

6. Jalankan query 1 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (500 total, Query took 0.0033 seconds.)
SELECT * FROM student
*****
```

Didapatkan waktu sebesar 0.0033 detik.

7. Jalankan query 2 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (367 total, Query took 0.0040 seconds.)
SELECT * FROM student WHERE tot_cred > 30
*****
```

Didapatkan waktu sebesar 0.0040 detik.

8. Jalankan query 3 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (367 total, Query took 0.0045 seconds.)
SELECT dept_name FROM student WHERE tot_cred > 30
*****
```

Didapatkan waktu sebesar 0.0045 detik.

9. Jalankan query 4 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (6089 total, Query took 0.0073 seconds.)
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id
*****
```

Didapatkan waktu sebesar 0.0073 detik.

10. Jalankan query 5 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (6089 total, Query took 0.0116 seconds.)
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
```

Didapatkan waktu sebesar 0.0116 detik.

11. Lakukan tuning indexing dengan memasukkan query seperti dibawah.

```
1 CREATE INDEX indexing ON takes(course_id) USING BTREE
```

```
1 CREATE INDEX indexing ON section(course_id) USING BTREE
```

12. Jalankan query 1 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (500 total, Query took 0.0032 seconds.)

```
SELECT * FROM `student`
```

Didapatkan waktu sebesar 0.0032 detik.

13. Jalankan query 2 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (367 total, Query took 0.0034 seconds.)

```
SELECT * FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0034 detik.

14. Jalankan query 3 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (367 total, Query took 0.0036 seconds.)

```
SELECT dept_name FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0036 detik.

15. Jalankan query 4 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (6089 total, Query took 0.0063 seconds.)

```
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id
```

Didapatkan waktu sebesar 0.0063 detik.

16. Jalankan query 5 (setelah tuning) dan catat waktunya.

Showing rows 0 - 24 (6089 total, Query took 0.0109 seconds.)

```
SELECT student_name, student_dept_name, takes_sec_id AS pengambil, 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
```

Didapatkan waktu sebesar 0.0109 detik.

D. Data 4

1. Buat database dengan nama rizqun4 untuk pengerjaan data keempat.

Create database

rizqun4 latin1_swedish_ci Create

2. Import DDL-MySQL.sql untuk menambahkan tabel-tabel pada database.

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in .[format].[compression]. Example: .sql.zip

Browse your computer: DDL-MySQL.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

3. Ubah tableGen.java dan sesuaikan dengan jumlah data yang diminta.

```
public static void main(String[] args) {  
    int classroom = 10;  
    int department = 10;  
    int course = 200;  
    int instructor = 50;  
    int teaches = 100;  
    int advisor = 700;  
    int student = 700;  
    int section = 20000;  
    int takes = 20000;  
    int prereq = 100;  
    int timeSlot = 10;  
    int i = 0, j = 0, r = 0, c = 0, x = 0, y = 0;  
    boolean tryValue = true;  
    String b = "", d = "", s = "";  
    fillArrays();  
}
```

4. Jika sudah sesuai, compile tableGen.java dengan menggunakan command prompt.

```
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>javac tableGen.java  
C:\Users\hp\Downloads\IF3144-1920-master\sql\tableGen>java tableGen
```

5. Maka akan terbentuk file all.sql, kemudian import all.sql tersebut kedalam database.

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in .[format].[compression]. Example: .sql.zip

Browse your computer: all.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

6. Jalankan query 1 (sebelum tuning) dan catat waktunya.

```
Showing rows 0 - 24 (700 total. Query took 0.0468 seconds.)  
SELECT * FROM student
```

Didapatkan waktu sebesar 0.0468 detik.

7. Jalankan query 2 (sebelum tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (524 total, Query took 0.1069 seconds.)  
  
SELECT * FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.1069 detik.

8. Jalankan query 3 (sebelum tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (524 total, Query took 0.0038 seconds.)  
  
SELECT dept_name FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0038 detik.

9. Jalankan query 4 (sebelum tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (2019700 total, Query took 0.0748 seconds.)  
  
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id
```

Didapatkan waktu sebesar 0.0748 detik.

10. Jalankan query 5 (sebelum tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (2019700 total, Query took 0.0692 seconds.)  
  
SELECT student_name, student_dept_name, takes_sec_id AS pengalihan, 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
```

Didapatkan waktu sebesar 0.0692 detik.

11. Lakukan tuning indexing dengan memasukkan query seperti dibawah.

```
1 CREATE INDEX indexing ON takes(course_id) USING BTREE  
  
1 CREATE INDEX indexing ON section(course_id) USING BTREE
```

12. Jalankan query 1 (setelah tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (700 total, Query took 0.0043 seconds.)  
  
SELECT * FROM 'student'
```

Didapatkan waktu sebesar 0.0043 detik.

13. Jalankan query 2 (setelah tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (524 total, Query took 0.0037 seconds.)  
  
SELECT * FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0037 detik.

14. Jalankan query 3 (setelah tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (524 total, Query took 0.0037 seconds.)  
  
SELECT dept_name FROM student WHERE tot_cred > 30
```

Didapatkan waktu sebesar 0.0037 detik.

15. Jalankan query 4 (setelah tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (2019700 total, Query took 0.0204 seconds.)  
  
SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id
```

Didapatkan waktu sebesar 0.0204 detik.

16. Jalankan query 5 (setelah tuning) dan catat waktunya.

```
✓ Showing rows 0 - 24 (2019700 total, Query took 0.0271 seconds.)  
  
SELECT student_name, student_dept_name, takes_seq_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
```

Didapatkan waktu sebesar 0.0271 detik.

BAB III

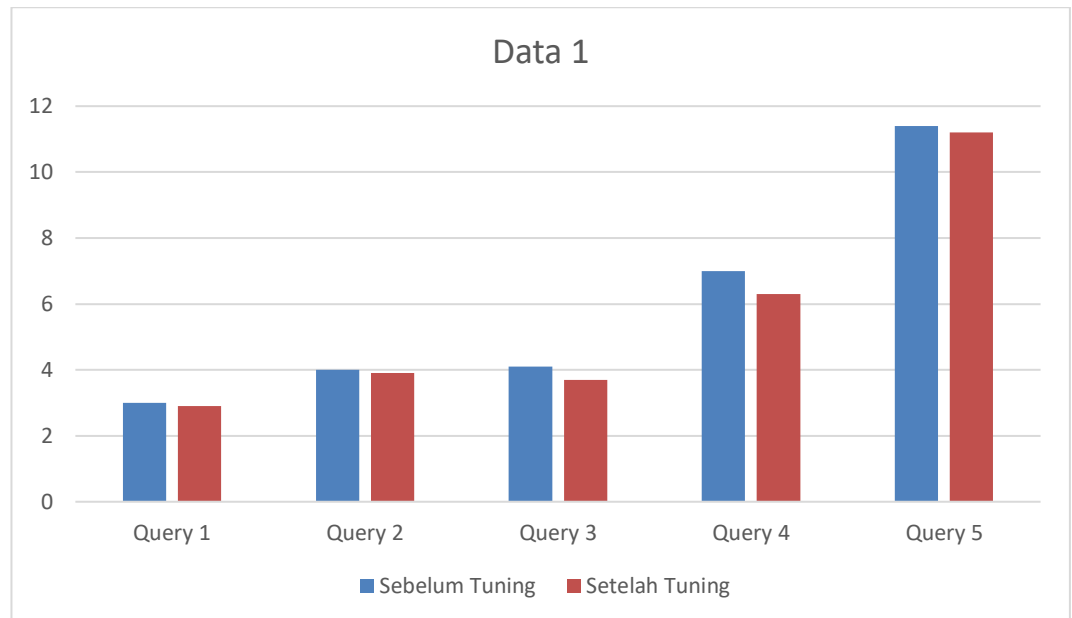
HASIL DAN PEMBAHASAN

3.1 Tabel Hasil

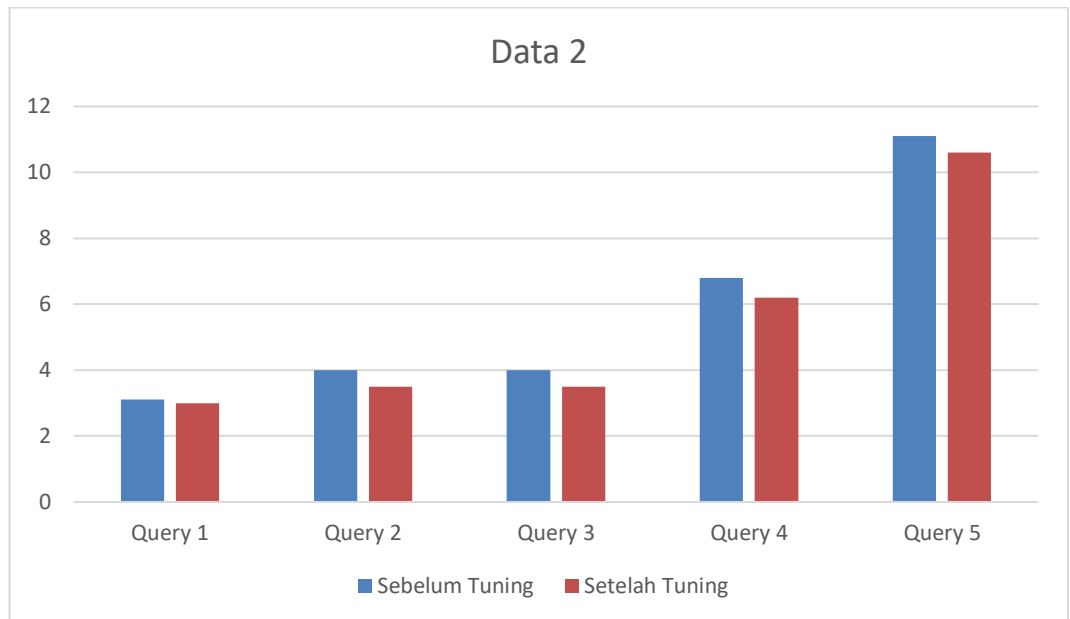
Data	Waktu sebelum Tuning (ms)					Waktu setelah Tuning (ms)				
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5
1	3	4	4.1	7	11.4	2.9	3.9	3.7	6.3	11.2
2	3.1	4	4	6.8	11.1	3	3.5	3.5	6.2	10.6
3	3.3	4	4.5	7.3	11.6	3.2	3.4	3.6	6.3	10.9
4	46.8	106.9	3.8	74.8	69.2	4.3	3.7	3.7	20.4	27.1
5	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-

3.2 Grafik Hasil

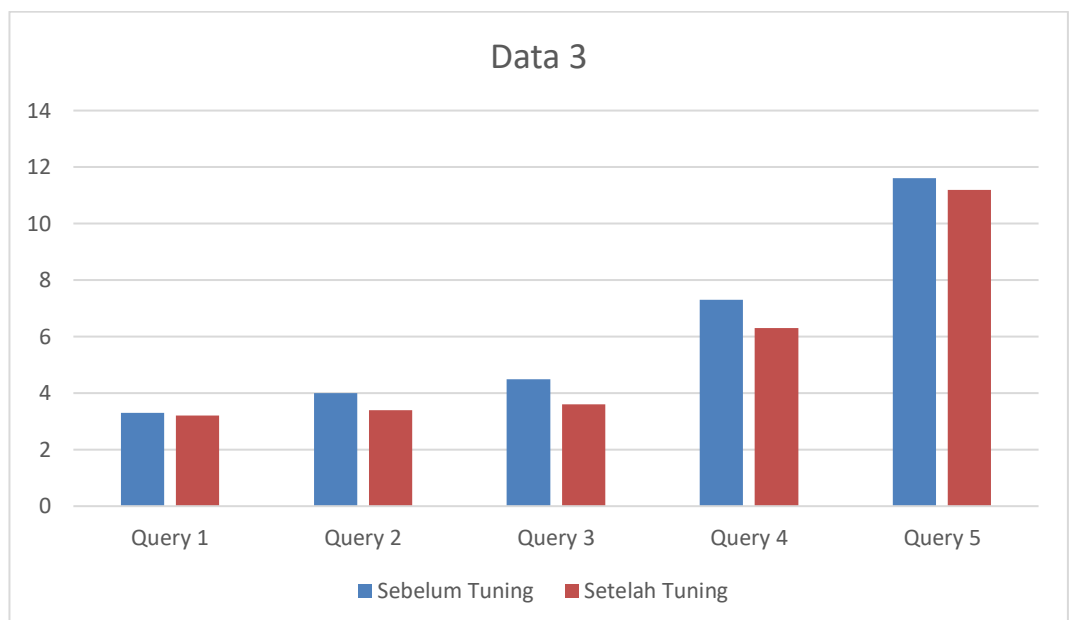
A. Data 1



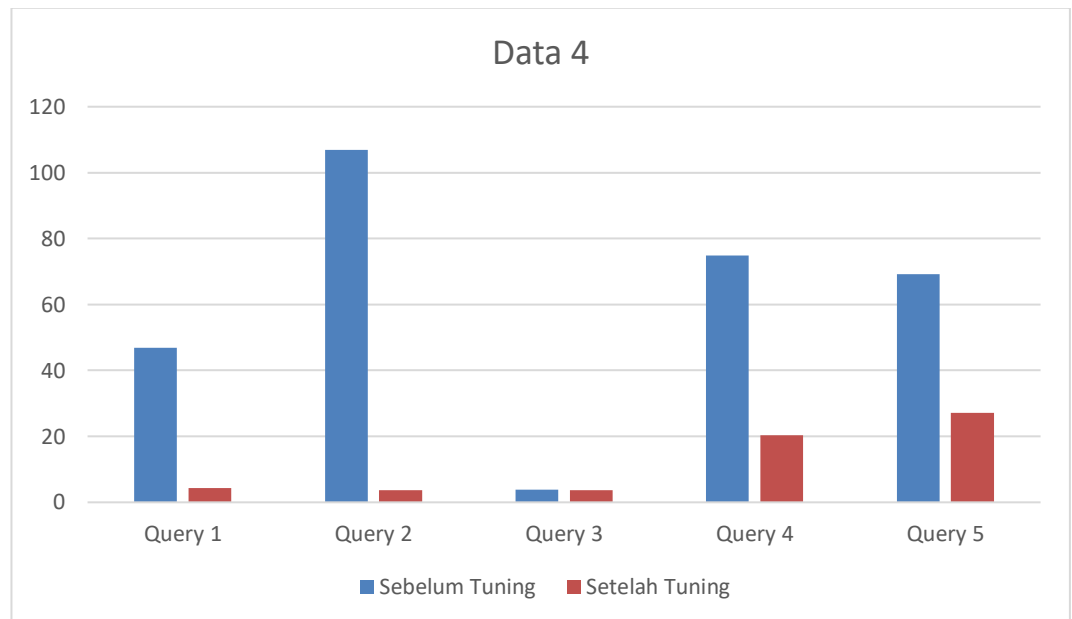
B. Data 2



C. Data 3



D. Data 4



3.3 Pembahasan Hasil

Dari hasil yang didapatkan dapat disimpulkan bahwa dengan melakukan tuning: indexing, dapat mempercepat waktu eksekusi pada basis data. Untuk data dengan jumlah yang tidak begitu banyak, seperti data 1, data 2, dan data 3, dapat dilihat bahwa perubahan waktu antara sebelum dan sesudah tuning tidak jauh berbeda, namun untuk data yang besar seperti data 4, dapat dilihat bahwa perbedaan waktunya sangat tinggi. Hal ini menandakan meskipun selisih waktunya tidak begitu jauh, namun dampaknya akan sangat terasa apabila digunakan pada basis data dengan jumlah data yang sangat besar. Hal ini tentunya cocok untuk diimplementasikan pada proyek atau perusahaan yang besar untuk menyimpan data-data dengan jumlah yang sangat banyak.

Untuk data 5, data 6, dan data 7, saya tidak dapat melakukan percobaan dikarenakan terjadi error ketika melakukan import untuk data ke-5. Proses melakukan import berjalan sangat lama dan tidak selesai-selesai hingga menyebabkan localhost saya tertutup dengan sendirinya. Saya telah mencoba memperbesar ukuran maksimum upload, membuat execution time menjadi tidak terbatas, dan menuliskan query secara langsung, namun data-5 tetap tidak dapat di import kedalam database.

DAFTAR PUSTAKA

<https://stackoverflow.com/questions/7680572/fatal-error-maximum-execution-time-of-300-seconds-exceeded>

<https://pojokprogrammer.net/content/performance-tuning-sederhana-di-mysql-menggunakan-index>

<https://stackoverflow.com/questions/3958615/import-file-size-limit-in-phpmyadmin>