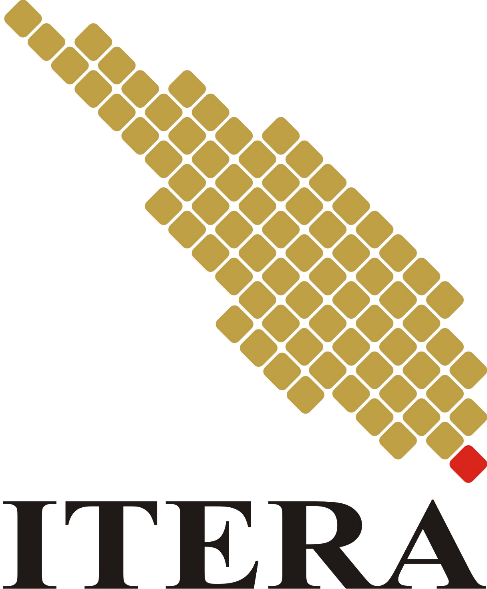
**TUGAS BESAR**

**MANAGEMEN BASIS DATA**

**TUNING DATABASE**

**Mata kuliah : Managemen Basis Data**



**Disusun oleh**

**Reza Octaviany 14117062**

**INFORMATICS ENGINEERING**

**INSTITUT TEKNOLOGI SUMATERA**

**2019**

**DAFTAR ISI**

[**BAB I PENDAHULUAN** 3](#_Toc27938446)

[1.1. Latar Belakang 3](#_Toc27938447)

[1.2. Tujuan 3](#_Toc27938448)

[**BAB III PERCOBAAN DAN ANALISIS DATA** 5](#_Toc27938449)

[3.1. Generate Data Awal 5](#_Toc27938450)

[3.2. Data yang dipakai 5](#_Toc27938451)

[3.3. Query yang digunakan 5](#_Toc27938452)

[3.4. Tuning 6](#_Toc27938453)

[3.5. Data Hasil Percobaan 6](#_Toc27938454)

[3.6. Analisis Percobaan 9](#_Toc27938455)

[**BAB IV KESIMPULAN** 10](#_Toc27938456)

[4.1. Kesimpulan 10](#_Toc27938457)

[**LAMPIRAN** 11](#_Toc27938458)

# **BAB I PENDAHULUAN**

## Latar Belakang

Basis data adalah kumpulan data mentah yang memiliki entitas dan relasi yang kemudian dapat diolah agar menjadi data yang memiliki makna. Basis data memiliki aplikasi untuk mengelola data-data, termasuk membuat, membaca, memperbarui, dan menghapus data yang ada. Pengelola basis data disebut Database Manajemen Sistem (DBMS).

Fitur-fitur yang diberikan oleh sebuah database berbagai macam, seperti keamanan, efisiens, kesediaan data, kelengkapan data, keamanan data dan *durability*. Sehingga database tersebut menjadi mudah digunakan, namun beberapa database yang bersifat gratis, tidak memiliki fitur yang lengkap, terutama pada hal efisiensi data.

Efisiensi pengambilan data pada database biasa disebut *tuning*. Efisiensi data pada database dapat dilakukan dengan cara indexing, yaitu *tuning* data. *Tuning* pada pengaturan atau konfigurasi DBMS juga dapat dilakukan. Maka kali ini, akan dilakukan percobaan untuk *tuning* database pada index data dan konfigurasi DBMS.

## Tujuan

Tujuan dari percobaan tugas besar Manajemen Basis Data ini adalah sebagai berikut:

* 1. Melengkapi tugas besar untuk matakuliah manajemen basis data.
  2. Memahami konsep dan cara melakukan tuning database dengan indexing.
  3. Memahami konsep dan cara melakukan tuning database dengan konfigurasi DBMS.
  4. Memahami perbedaan waktu yang diperoleh masing-masing komputer, sesuai dengan spesifikasi komputer.

**BAB II**

**LANDASAN TEORI**

## 2.1. Database Management System (DBMS)

DBMS adalah koleksi data yang besar yang terintegrasi, yang memodelkan data nyata, yang memiliki entitas dan relasi. Yang disimpan di paket software dan me-*manage* database.

## 2.2. Tuning Database

Tuning database adalah aktivitas membuat database bekerja lebih cepat. Artinya, lebih besar throughput, akan lebih baik, dengan satuan througput per-second.

## 2.3. MySQL

MySQL adalah sebuah database management yang memberikan jasa pengolahan database yang gratis.

## 2.4. NetBeans

Netbeans adalah IDE tools untuk mengolah bahasa java yang dibutuhkan pada praktikum kali ini. Untuk generate data otomatis pada database.

## 2.5. Spesifikasi Laptop

Prosesor : Intel® Core ™ i7-7700HQ CPU @ 2.80GHz 2.81GHz

RAM : 16.0 GB

OS : Windows 10 Pro N

# **BAB III PERCOBAAN DAN ANALISIS DATA**

## 3.1. Generate Data Awal

Data awal yang digunakan diperoleh dari *generate* data, dengan menggunakan file tableGen.zip yang berisikan file java yang telah diberikan. Menambahkan konfigurasi pada *system environtment*, kemudian melakukan *execute* data melalui file java, menggunakan perintah:

javac tablegen.java

Untuk mendapatkan file tablegen dengan ekstensi class.

javac tableGen

Untuk mendapatkan file all.sql

## 3.2. Data yang dipakai

Data yang dipakai adalah sebagai berikut:

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

## 3.3. Query yang digunakan

Query yang digunakan adalah sebagai berikut:

1. SELECT \* FROM student
2. SELECT \* FROM student WHERE tot\_cred > 30;
3. SELECT `name`, department 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

## 3.4. Tuning

Tuning database yang digunakan terbagi menjadi dua bagian, yaitu:

1. Tuning dengan mengubah konfigurasi *my.ini* yaitu dengan mengubah nilai-nilai antara lain:
   1. innodb\_buffer\_pool\_size dari 16M menjadi 2000M
   2. innodb\_log\_file\_size, size dari 5M menjadi 100M
   3. innodb\_log\_buffer\_size, size dari 8M menjadi 100M
2. Tuning dengan melakukan indexing pada beberapa kolom table:
   1. Data name di table student
   2. Data tot\_cred di table student
   3. Data grade di table takes
   4. Data time\_slot\_id di table section

## 3.5. Data Hasil Percobaan

Berikut ini adalah data yang didapatkan sebelum dan setelah melakukan ujicoba tuning pada index table dan tuning melalui konfigurasi.

1. Query = SELECT \* FROM student

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Waktu Sebelum Tunning (ms)** | **Waktu Sesudah Tuning (ms)** | |
| **konfigurasi** | **indexing** |
| advisor = 100, student = 100, section = 200,takes = 200 | 0.016 | 0.092 | 0.015 |
| advisor = 200, student = 200, section = 400,takes = 400 | 0.015 | 0.012 | 0.014 |
| advisor = 500, student = 500, section = 1000,takes = 1000 | 0.021 | 0.019 | 0.022 |
| advisor = 700, student = 700, section = 20000,takes = 20000 | 0.071 | 0.068 | 0.071 |
| advisor = 1000, student = 1000, section = 100000,takes = 1000000 | 0.101 | 0.081 | 0.094 |
| advisor = 1800, student = 1800, section = 180000,takes = 1800000 | 0.502 | 0.48 | 0.482 |
| advisor = 10000, student = 10000, section = 30000000,takes = 30000000 | 2.512 | 2.34 | 2.418 |

1. Query = SELECT \* FROM student WHERE tot\_cred > 30;

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Waktu Sebelum Tunning (ms)** | **Waktu Sesudah Tuning (ms)** | |
| **konfigurasi** | **indexing** |
| advisor = 100, student = 100, section = 200,takes = 200 | 0.161 | 0.091 | 0.15 |
| advisor = 200, student = 200, section = 400,takes = 400 | 0.245 | 0.164 | 0.141 |
| advisor = 500, student = 500, section = 1000,takes = 1000 | 0.352 | 0.225 | 0.245 |
| advisor = 700, student = 700, section = 20000,takes = 20000 | 1.108 | 0.911 | 1.020 |
| advisor = 1000, student = 1000, section = 100000,takes = 1000000 | 2.001 | 1.921 | 1.952 |
| advisor = 1800, student = 1800, section = 180000,takes = 1800000 | 3.431 | 3.202 | 3.402 |
| advisor = 10000, student = 10000, section = 30000000,takes = 30000000 | 10.412 | 10.212 | 10.332 |

1. Query = SELECT `name`, department FROM student WHERE tot\_cred > 30;

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Waktu Sebelum Tunning (ms)** | **Waktu Sesudah Tuning (ms)** | |
| **konfigurasi** | **indexing** |
| advisor = 100, student = 100, section = 200,takes = 200 | 0.125 | 0.078 | 0.094 |
| advisor = 200, student = 200, section = 400,takes = 400 | 0.21 | 0.151 | 0.192 |
| advisor = 500, student = 500, section = 1000,takes = 1000 | 0.188 | 0.135 | 0.159 |
| advisor = 700, student = 700, section = 20000,takes = 20000 | 0.548 | 0.512 | 0.523 |
| advisor = 1000, student = 1000, section = 100000,takes = 1000000 | 1.243 | 1.201 | 1.232 |
| advisor = 1800, student = 1800, section = 180000,takes = 1800000 | 2.411 | 2.3 | 2.401 |
| advisor = 10000, student = 10000, section = 30000000,takes = 30000000 | 20.71 | 19.274 | 19.726 |

1. Query = SELECT \* FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course\_id = section.course\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Waktu Sebelum Tunning (ms)** | **Waktu Sesudah Tuning (ms)** | |
| **konfigurasi** | **indexing** |
| advisor = 100, student = 100, section = 200,takes = 200 | 0.875 | 0.442 | 0.537 |
| advisor = 200, student = 200, section = 400,takes = 400 | 1.747 | 1.387 | 16.35 |
| advisor = 500, student = 500, section = 1000,takes = 1000 | 5.681 | 3.691 | 4.129 |
| advisor = 700, student = 700, section = 20000,takes = 20000 | 273.898 | 272.401 | 273.502 |
| advisor = 1000, student = 1000, section = 100000,takes = 1000000 | 371.591 | 361.391 | 262.844 |
| advisor = 1800, student = 1800, section = 180000,takes = 1800000 | 543.847 | 528.017 | 539.815 |
| advisor = 10000, student = 10000, section = 30000000,takes = 30000000 | 1243.816 | 1025.014 | 1110.527 |

1. Query = 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

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Waktu Sebelum Tunning (ms)** | **Waktu Sesudah Tuning (ms)** | |
| **konfigurasi** | **indexing** |
| advisor = 100, student = 100, section = 200,takes = 200 | 0.663 | 0.522 | 0.635 |
| advisor = 200, student = 200, section = 400,takes = 400 | 1.251 | 1.225 | 1.382 |
| advisor = 500, student = 500, section = 1000,takes = 1000 | 3.708 | 3.347 | 3.562 |
| advisor = 700, student = 700, section = 20000,takes = 20000 | 141.066 | 134.06 | 137.061 |
| advisor = 1000, student = 1000, section = 100000,takes = 1000000 | 345.008 | 324.098 | 331.673 |
| advisor = 1800, student = 1800, section = 180000,takes = 1800000 | 453.842 | 433.047 | 435.159 |
| advisor = 10000, student = 10000, section = 30000000,takes = 30000000 | 1142.861 | 1005.011 | 1010.527 |

## 3.6. Analisis Percobaan

1. Pada percobaan query diperoleh bahwa query yang paling cepat adalah query pertama, karena data yang diminta paling sedikit, dan tidak ada data yang di gabungkan (JOIN). Selain itu Query pertama juga memiliki perintah yang paling simple. Query yang paling banyak memakan waktu adalah Query ke 4, karena ada 2 tabel yang di gabbungkan (JOIN).
2. Pada data yang ada, data yang paling cepat di eksekusi adalah data pertama, karena memiliki data paling sedikit, begitu pula sebaliknya query dengan data terbanyak (query ke-5) memakan waktu leebih lama untuk di eksekusi.
3. Waktu sesudah tuning pada konfigurasi DBMS lebih baik daripada tuning dengan indexing data. Karena, tuning dengan konfigurasi melibatkan *cache,* yang merupakan memori tercepat yang ada, dan pada konfigurasi DBMS tuning dilakukan untuk menambah besar cache yang ada pada DBMS.
4. Penulis telah membandingkan eksekusi query yang dilakukan dengan orang lain, bahwa spesifikasi laptop ternyata berpengaruh kepada eksekusi query.

# **BAB IV PENUTUP**

## Kesimpulan

Kesimpulan yang dapat diperoleh pada analisis data di atas adalah sebagai berikut :

1. Semakin banyak penggabungan data pada tabel (JOIN) semakin banyak waktu yang dihabiskan untuk meng-eksekusi query
2. Semakin banyak data yang ada pada tabel maka semakin lama eksekusi yang akan dijalankan
3. Semakin baik tuning yang dilakukan semakin cepat eksekusi berlangsung.
4. Semakin baik spesifikasi laptop semakin cepat eksekusi terjadi.

# **LAMPIRAN**

DATA TABEL 3

