TUGAS BESAR

MANAGEMEN BASIS DATA

TUNING DATABASE

DISUSUN OLEH

Muhammad Taufiq Hidayat / 14116162

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2019

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. 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).

Database sendiri memiliki proses *input, update, delete,* dan *read,* dimana semua data tersebut diakses dengan menggunakan query, dan terdapat waktu eksekusi. Ketika, database tersebut tidak disiapkan dengan baik, maka waktu eksekusi query akan terbilang cukup lama. Oleh karena itu, terdapat database tuning yang digunakan untuk meningkatkan efisiensi dan kecepatan proses eksekusi query.

1.2. Tujuan

Adapun tujuan dari tugas besar managemen basis data, diantara lain :

1. Melengkapi tugas besar Managemen Basis Data
2. Memahami konsep tuning dengan indexing dan konfigurasi

BAB II
Landasan Teori

2.1. Tuning Indexing

Tuning Indexing sendiri jika digambarkan sama seperti halnya buku, dimana buku memiliki daftar isi, sehingga ketika kita ingin mencari sesuatu dapat melihat index dan langsung menemukan pada halaman berapa sesuatu yang kita cari tersebut. Dan tentu saja akan ada hal yang dikorbankan yaitu penambahan tempat memory untuk menyimpan index nya.

Pada database sendiri cara untuk melakukan indexing adalah memberikan index pada kolom yang sering dijadikan parameter dalam *where clause*.

2.2. Tuning Konfigurasi DBMS

Tuning database pada level konfigurasi dimana mengubah beberapa settingan dari database untuk menselaraskan dengan *hardware* yang digunakan.

2.3. Spesifikasi Laptop

- Processor : Core I7 gen 6
- RAM : 16GB
- OS : Elementary OS (Linux)

BAB III

PERCOBAAN

3.1. Persiapan data

Pertama buat database dengan nama dbms1 sampai dengan dbms7, dikarenakan akan ada 7 data yang akan kita proses. Setelah database dibuat, sekarang kita lakukan import skema database dengan memanggil DDL-MySQL.sql. Gunakan perintah:

```
➜  ~ mysql -ufiq -pfiq dbms1 < ~/Documents/IF3144-1920/sql/DDL-MySQL.sql
```

Lakukan perintah diatas dimana **dbms1** diganti dengan setiap nama database (dbms2, dbms3, ..., dbms7).

Setelah ini lakukan generating data random dengan tableGen.Java yang telah disediakan, sebelum di compile sesuaikan dulu jumlah max data yang diperlukan, dan banyak data yang akan digenerate untuk setiap tabel nya. Cara melakukan compile adalah jalankan command dibawah ini:

```
➜  ~ javac Documents/1/tableGen.java
➜  ~ java Documents.1.tableGen
```

Dimana perintah utamanya adalah

javac file.java dan *java file*

Setelah setiap data di generate, akan menghasilkan file all.sql di dalam folder sql. Berikutnya kita perlu melakukan import lagi untuk setiap isi dari dbms1 sampai dengan dbms7. caranya adalah sebagai berikut :

```
~ mysql -ufiq -pfiq dbms1 < Documents/1/sql/all.sql
```

Lakukan sampai dbms ke 7.

3.2. Tuning

Tuning yang dilakukan untuk meakukan indexing dan konfigurasi pada DBMS adalah sebagai berikut:

Konfigurasi MySQL

- Key_buffer_size dari 16M menjadi 2000M
- Thread_cache_size, size dari 16M menjadi 100M
- Thread_cache_size, size dari 8 menjadi 12

Indexing

- Data name di table student
- Data tot_cred di table student
- Data grade di table takes
- Data time_slot_id di table section

Cara melakukan konfigurasi pada sistem operasi Linux adalah sebagai berikut:

1. Edit file pada /etc/mysql/mariadb.conf.d
2. Akan ada 4 file yang muncul, lalu edit pada bagian 50-server.cnf

Dan langkah untuk melakukan indexing adalah dengan menjalankan command dibawah ini:

Create index **nama_index** on **nama_table**(**nama_kolom**)

3.3. Hasil

SELECT * FROM student			
DATA	**Waktu Sebelum Tuning (ms)**	**Waktu Sesudah Tuning (ms)**	
		Konfigurasi	**Indexing**
advisor = 100, student = 100, section = 200,takes = 200	0.095	0.091	0.015
advisor = 200, student = 200, section = 400,takes = 400	0.85	0.050	0.017
advisor = 500, student = 500, section =	0.76	0.54	0.34

1000,takes = 1000			
advisor = 700, student = 700, section = 20000,takes = 20000	2.47	1.74	1.53
advisor = 1000, student = 1000, section = 100000,takes = 1000000	2.52	2.32	1.57
advisor = 1800, student = 1800, section = 180000,takes = 1800000	2.72	2.35	2.43
advisor = 10000, student = 10000, section = 30000000,takes = 30000000	2.81	2.62	2.57

Query = SELECT * FROM student WHERE tot_cred > 30;			
DATA	**Waktu Sebelum Tuning (ms)**	**Waktu Sesudah Tuning (ms)**	
		Konfigurasi	**Indexing**
advisor = 100, student = 100, section = 200,takes = 200	0.093	0.083	0.071
advisor = 200, student = 200, section = 400,takes = 400	0.232	0.174	0.153
advisor = 500, student = 500, section = 1000,takes = 1000	0.473	0.357	0.219
advisor = 700, student = 700, section = 20000,takes = 20000	0.83	0.801	0.759
advisor = 1000,	1.921	1.937	1.811

student = 1000, section = 100000,takes = 1000000			
advisor = 1800, student = 1800, section = 180000,takes = 1800000	3.631	3.343	3.17
advisor = 10000, student = 10000, section = 30000000,takes = 30000000	10.193	9.937	9.251

Query = SELECT `name`, department FROM student WHERE tot_cred > 30;			
DATA	**Waktu Sebelum Tuning (ms)**	**Waktu Sesudah Tuning (ms)**	
		Konfigurasi	**Indexing**
advisor = 100, student = 100, section = 200,takes = 200	0.103	0.081	0.077
advisor = 200, student = 200, section = 400,takes = 400	0.172	0.129	0.158
advisor = 500, student = 500, section = 1000,takes = 1000	0.536	0.526	0.325
advisor = 700, student = 700, section = 20000,takes = 20000	1.452	1.236	1.983
advisor = 1000, student = 1000, section = 100000,takes = 1000000	1.731	1.534	1.42
advisor = 1800, student = 1800, section =	2.527	2.345	2.138

180000,takes = 1800000			
advisor = 10000, student = 10000, section = 30000000,takes = 30000000	24.213	20.124	19.475

Query = SELECT * FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course_id = section.course_id			
DATA	**Waktu Sebelum Tuning (ms)**	**Waktu Sesudah Tuning (ms)**	
		Konfigurasi	**Indexing**
advisor = 100, student = 100, section = 200,takes = 200	0.738	0.912	0.642
advisor = 200, student = 200, section = 400,takes = 400	1.924	1.673	1.746
advisor = 500, student = 500, section = 1000,takes = 1000	7.631	7.214	6.825
advisor = 700, student = 700, section = 20000,takes = 20000	253.2	245.4	241.6
advisor = 1000, student = 1000, section = 100000,takes = 1000000	351.5	344.6	338.3
advisor = 1800, student = 1800, section = 180000,takes = 1800000	559.2	539.1	532.4
advisor = 10000, student = 10000, section = 30000000,take	1635	1621	1583

s = 30000000			

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 Tuning (ms)**	**Waktu Sesudah Tuning (ms)**	
		Konfigurasi	**Indexing**
advisor = 100, student = 100, section = 200,takes = 200	0.532	0.482	0.356
advisor = 200, student = 200, section = 400,takes = 400	1.326	1.183	1.236
advisor = 500, student = 500, section = 1000,takes = 1000	3.832	3.635	3.521
advisor = 700, student = 700, section = 20000,takes = 20000	139.7	134.8	127.3
advisor = 1000, student = 1000, section = 100000,takes = 1000000	150.7	152.5	146.8
advisor = 1800, student = 1800, section = 180000,takes = 1800000	424.2	422.5	416.3
advisor = 10000, student = 10000,	1642	1382	1186

section = 30000000,takes = 30000000			

3.4. Analisis

1. Query yang memiliki JOIN akan lebih memakan waktu eksekusi lebih lama
2. Query yang memiliki banyak *Where clause* akan memiliki eksekusi yang lama
3. Query yang memiliki JOIN dan *Where clause* secara bersamaan akan berjalan sangat lama dan memakan memory sangat besar
4. Secara rata-rata tuning dengan indexing lebih optimal dibandingan konfigurasi saja
5. Konfigurasi bisa jadi berbahaya ketika terlalu banyak mengalokasikan memory atau terlalu sedikit, bisa membuat *server* hang
6. Data bisa jadi tidak akurat dikarenakan pada saat eksekusi setiap query kondisi sistem tidak hanya menggunakan untuk *dbms.*