**LAPORAN TUGAS BESAR  
MANAJEMEN BASIS DATA RA**



Disusun oleh :

Alya Khairunnisa Rizkita 14117095

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI SUMATERA**

**2019**

**DAFTAR ISI**

[**ISI** 3](#_Toc27986827)

[**1.** **STUDI LITERATUR** 3](#_Toc27986828)

[**2.** **DESKRIPSI PERCOBAAN** 4](#_Toc27986829)

[**DAFTAR PUSTAKA** 8](#_Toc27986830)

# **ISI**

## **STUDI LITERATUR**

1. Tunning Indexing

Indeks adalah solusi untuk banyak masalah kinerja, tetapi memiliki terlalu banyak indeks pada tabel yang sering diperbarui dapat menimbulkan biaya tambahan karena SQL Server melakukan pekerjaan ekstra untuk menjaga agar indeks tetap terbaru selama memasukkan / memperbarui / menghapus operasi. Jadi mesin database SQL Server membutuhkan lebih banyak waktu ketika memperbarui data dalam tabel berdasarkan jumlah dan kompleksitas indeks. Juga, pemeliharaan indeks nance dapat meningkatkan penggunaan CPU dan I / O, yang dapat merusak kinerja dalam a sistem intensif menulis. Kita harus menghapus duplikat dan indeks yang berlebihan seperti mereka adalah menguras sumber daya sistem.

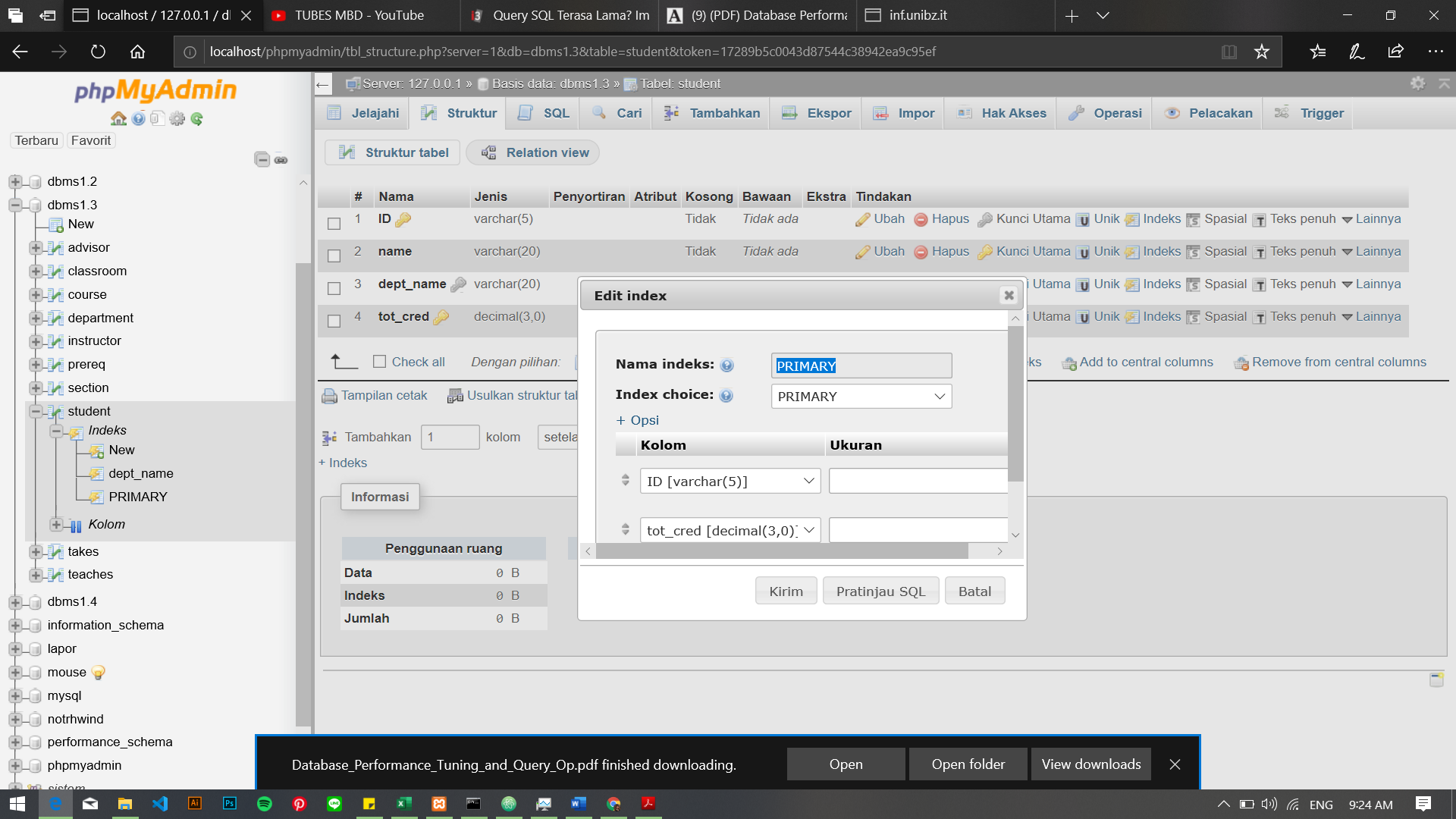
1. Tunning DBMS

Memori mempengaruhi kinerja SQL Server lebih dari perangkat keras lainnya. Oleh karena itu, perlu untuk memantau penggunaan memori secara teratur pada sistem SQL Server memastikan bahwa persentase memori yang tersedia lebih tinggi dari 20%. Jika pengguna mengalami masalah kinerja dan persentase memori yang tersedia turun di bawah ini 20%, maka masalahnya insuf alokasi memori cient. Amati kehidupan halaman rata-rata penghitung kinerja harapan dan pastikan selalu di atas 300 detik (5 menit). Apa pun yang kurang menunjukkan desain indeks yang buruk mengarah ke peningkatan input / output disk (I / O) dan penggunaan memori yang kurang efektif atau kekurangan memori yang sebenarnya. Monitor tingkat paging pada sistem SQL Server, dan pastikan mereka secara teratur di atas 1.000 halaman per detik. Biasanya, transaksi OLTP kecil tidak memerlukan memori besar hibah. Apa pun yang lebih besar dari hibah memori nol untuk transaksi OLTP menunjukkan kehabisan memori dalam sistem SQL Server [ 12 ] Salah satu cara untuk mengatasi kemacetan memori adalah dengan dan proses intensif memori, yang dapat digunakan untuk mengidentifikasi potensi masalah aplikasi seperti kebocoran memori. Satu juga dapat meninjau permintaan untuk mengoptimalkan kinerja untuk mengkonsumsi lebih sedikit memori. Pendekatan lain adalah meningkatkan skala lingkungan SQL Server dengan menambahkan lebih banyak fisik Penyesuaian Kinerja Database dan Optimalisasi Permintaan 5 memori (RAM) ke SQL Server. Meningkatkan skala biasanya merupakan pendekatan yang baik untuk mengatasi setiap hambatan kinerja yang terkait dengan memori.

## **DESKRIPSI PERCOBAAN**

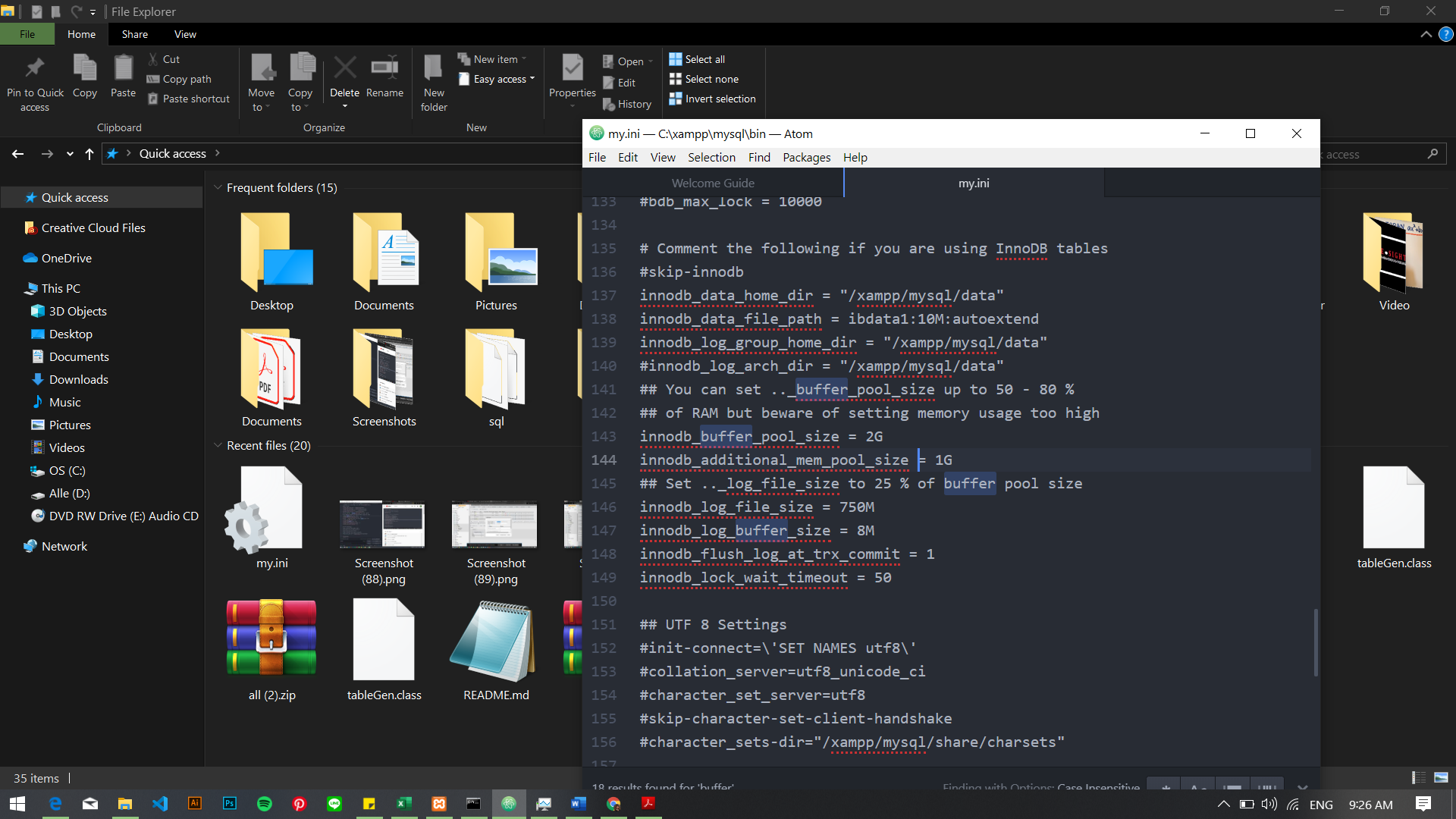
1. Tunning Indexing

Pada percobaan kali ini indecing diatur sedemikian rupa supaya pada saat melakukan pengoprasian query tidak memakan waktu yang terlalu lama. Untuk sekarang indexing sudah terdaftar pada setiap table yang ada, namun ada satu table yang membutuhkan tambahan index yaitu table *student* yang terdapat tot\_cre yang membutuhkan indexing. Sehingga pada tahap tunning indexing kita hanya menambahkan index baru untuk tot\_cre pada table student.



1. Tunnung Setting Configuration DBMS

Pada percobaan ini pengaturan terhadap konfigurasi DBMS dapat dilakukan pada file my.ini yang dapat dilihat pada XAMPP. Untuk itu kita harus merubah beberapa pengaturan pemakaian memori pada saat melakukan eksekusi query pada data base. Aturlah besaran memori yang akan dipakai dengan besar memory yang ada pada komputer / laptop masing-masing. Pada kesempatan ini saya mengatur penggunaan memori seperti berikut.



1. Hasil dan Pembahasan

Setelah melakukan tunning pada indexing dan konfigurasi dbms dapat dilihat perbandingan waktu yang dihabiskan pada setiap query.

Query yang di gunakan pada pengujian kali ini terdapat :

1. SELECT \* FROM student;
2. SELECT \* FROM student WHERE tot\_cred > 30;
3. SELECT \* FROM takes JOIN student ON takes.ID = student.ID JOIN section ON takes.course\_id = section.course\_id;
4. 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;

Berikut waktu yang dihabiskan pada setiap pemrosesan query.

1. Waktu sebelum tunning.

|  |  |
| --- | --- |
| **NO** | **Data** |
| 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 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO** | **Waktu Sebelum Tunning (ms)** | | | | |
| 1 | Q1 | Q2 | Q3 | Q4 | Q5 |
| 2 | 0.0004004 | 0.0006863 | 0.0003303 | 0.0056381 | 0.0404308 |
| 3 | 0.0004052 | 0.000397 | 0.0003676 | 0.0028853 | 0.0022224 |
| 4 | 0.0281002 | 0.0007053 | 0.0004803 | 0.0667592 | 0.0809309 |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
|  |  |  |  |  |  |

1. Waktu Sesudah Tunning Index (ms)

|  |  |
| --- | --- |
| **NO** | **Data** |
| 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 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Waktu Sesudah Tunning Index (ms)** | | | | |
| Q1 | Q2 | Q3 | Q4 | Q5 |
| 1 | 0.0389889 | 0.0004518 | 0.0002113 | 0.0003564 | 0.0026366 |
| 2 | 0.0003741 | 0.0004981 | 0.0006559 | 0.0063194 | 0.0047197 |
| 3 | 0.0012063 | 0.0011438 | 0.0008417 | 0.0143156 | 0.062556 |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |

1. Waktu sesudah tunning DBMS (ms)

|  |  |
| --- | --- |
| **NO** | **Data** |
| 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 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Waktu Sesudah Tunning DBMS (ms)** | | | | |
| Q1 | Q2 | Q3 | Q4 | Q5 |
| 1 | 0.0005937 | 0.000669 | 0.000357 | 0.0013469 | 0.0013754 |
| 2 | 0.0006145 | 0.0004059 | 0.0003519 | 0.0030586 | 0.0034165 |
| 3 | 0.0005773 | 0.0011652 | 0.000431 | 0.0138692 | 0.0152707 |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |

# **DAFTAR PUSTAKA**

<https://www.i-3.co.id/2016/10/07/index-pada-database/>

<https://www.academia.edu/37005404/Database_Performance_Tuning_and_Query_Optimization?auto=download>