OPTIMASI & QUERY PERFORMANCE
BASIS DATA II

Disusun Oleh :

Harits Putra Junaidi 230411100003

Achmad Lutfi Madhani 230411100059

Danendra Mahardhika 230411100086

Ahmad Ubaydir Rohman 230411100116

Moh Naufal Thariq 230411100142

Seinal Arifin 230411100149

Dosen Pengampu :

Nama : Moch Kautsar Sophan, S.Kom, M.MT

NIP : 197707132002121004

PRODI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

2025

Topik: Analisis Efektivitas Query dan Indeks menggunakan Studi Kasus SIAKAD

Instruksi Umum

1. Gunakan database SIAKAD yang sudah berisi data mahasiswa, krs, nilai, dan mata kuliah.

2. Buat dan eksekusi 3 jenis query sebagaimana dijelaskan di bawah.

3. Setiap query dieksekusi minimal 10 kali, dan hasil waktu eksekusinya dicatat dalam bentuk tabel.

4. Gunakan cara pengukuran waktu eksekusi query dengan metode berikut:

```
-- Waktu mulai
SET @start := CURRENT_TIMESTAMP(6);

-- Query yang diukur (bebas), berikut contoh saja
SELECT * FROM thutang ORDER BY nodokumen; -- Ganti dengan querymu

-- Waktu akhir
SET @end := CURRENT_TIMESTAMP(6);

-- Hitung durasi
SELECT TIMEDIFF(@end, @start) AS waktu_detik,
    TIME_TO_SEC(TIMEDIFF(@end, @start)) AS waktu_dlm_detik,
    (@end - @start) AS beda_raw;
```

5. Catat hasil waktu dalam tabel hasil eksekusi, lalu buat analisis performa dan kemungkinan optimasi (misal: penggunaan indeks).

Keterangan Tools yang saya pakai:

Adminer.php agar mempermudah akses Database online saya

Tugas Spesifik

1. Menampilkan IPK Mahasiswa Angkatan 2019 per Prodi Keluaran yang diharapkan: angkatan, prodi, nim, ipk Ulangi 10x, catat waktu.

   Query :

```
SET @start := CURRENT_TIMESTAMP(6);

SELECT

    m.tahun_masuk AS angkatan,

    p.nama_prodi AS prodi,

    m.nim,

    fn_hitung_ipk(m.id_mahasiswa) AS ipk

FROM

    mahasiswa m

JOIN

    program_studi p ON m.id_prodi = p.id_prodi

WHERE

    m.tahun_masuk = 2019

    AND m.status = 'Aktif'

ORDER BY

    p.nama_prodi, m.nim;

SET @end := CURRENT_TIMESTAMP(6);

SELECT

    TIMEDIFF(@end, @start) AS waktu_detik,

    TIME_TO_SEC(TIMEDIFF(@end, @start)) AS waktu_dlm_detik,

    (@end - @start) AS beda_raw;
```

Penjelasan Query

Skrip SQL ini digunakan untuk menampilkan data mahasiswa angkatan 2019 yang masih berstatus aktif, beserta program studinya dan nilai IPK yang dihitung melalui fungsi khusus. Selain itu, skrip ini juga mengukur durasi waktu yang dibutuhkan untuk menjalankan perintah SELECT tersebut, dengan tingkat ketelitian hingga mikrodetik.

Penjelasan setiap bagian skrip adalah sebagai berikut:

Inisialisasi Waktu Awal Eksekusi

```
SET @start := CURRENT_TIMESTAMP(6);
```

1. Baris ini berfungsi untuk menyimpan waktu saat proses pengambilan data dimulai. CURRENT_TIMESTAMP(6) akan menghasilkan waktu lengkap hingga enam digit di belakang koma (mikrodetik), dan nilainya disimpan dalam variabel @start.

Pengambilan Data Mahasiswa

```
SELECT
    m.tahun_masuk AS angkatan,
    p.nama_prodi AS prodi,
    m.nim,
    fn_hitung_ipk(m.id_mahasiswa) AS ipk
FROM
    mahasiswa m
JOIN
    program_studi p ON m.id_prodi = p.id_prodi
WHERE
    m.tahun_masuk = 2019
    AND m.status = 'Aktif'
ORDER BY
    p.nama_prodi, m.nim;
```

2. Bagian ini melakukan pengambilan data dari tabel mahasiswa yang di-*join* dengan tabel program_studi berdasarkan id_prodi . Data yang ditampilkan mencakup tahun masuk mahasiswa (sebagai angkatan), nama program studi, NIM, dan IPK yang dihitung menggunakan fungsi fn_hitung_ipk. Data yang ditampilkan dibatasi

hanya untuk mahasiswa angkatan 2019 dengan status aktif, dan hasilnya diurutkan berdasarkan nama program studi, lalu NIM.

```
SET @end := CURRENT_TIMESTAMP(6);
```

3. Setelah proses SELECT selesai dijalankan, waktu saat eksekusi berakhir disimpan ke dalam variabel @end.

Penghitungan Lama Eksekusi

```
SELECT
  TIMEDIFF(@end, @start) AS waktu_detik,
  TIME_TO_SEC(TIMEDIFF(@end, @start)) AS waktu_dlm_detik,
  (@end - @start) AS beda_raw;
```

4. Pada bagian ini, dihitung selisih waktu antara @end dan @start untuk mengetahui durasi proses pengambilan data. TIMEDIFF menampilkan selisih waktu dalam format jam-menit-detik, TIME_TO_SEC mengubahnya menjadi jumlah detik total, dan ekspresi @end - @start memberikan hasil selisih waktu dalam format mentah.

Tabel Output:

Percobaan Ke	waktu_detik	Waktu_dalam_detik	beda_raw
1	17.214373	17	0
2	37.639441	37	0
3	4.719192	4	0
4	12.763959	12	0
5	28.723040	28	0
6	20.536994	20	0
7	47.997543	47	0
8	5.451147	5	0
9	5.502141	5	0
10	6.770048	6	0

2. Menampilkan Transkrip Mahasiswa berdasarkan NIM

Input: NIM tertentu

Keluaran yang diharapkan:

nama_matkul, sks, nilai_angka, nilai_huruf, bobot, hasil kali bobot dan sks

Ulangi 10x, catat waktu.

Query:

```
SET @start := CURRENT_TIMESTAMP(6);
SET @nim = 2019010340;
SELECT
    mk.nama_mata_kuliah AS nama_matkul,
    mk.sks,
    n.nilai_akhir AS nilai_angka,
    n.grade AS nilai_huruf,
    CASE
        WHEN n.grade = 'A' THEN 4.00
        WHEN n.grade = 'A-' THEN 3.70
        WHEN n.grade = 'B+' THEN 3.30
        WHEN n.grade = 'B' THEN 3.00
        WHEN n.grade = 'B-' THEN 2.70
        WHEN n.grade = 'C+' THEN 2.30
        WHEN n.grade = 'C' THEN 2.00
        WHEN n.grade = 'D' THEN 1.00
        WHEN n.grade = 'E' THEN 0.00
        ELSE 0
    END AS bobot,
    CASE
        WHEN n.grade = 'A' THEN 4.00 * mk.sks
        WHEN n.grade = 'A-' THEN 3.70 * mk.sks
        WHEN n.grade = 'B+' THEN 3.30 * mk.sks
        WHEN n.grade = 'B' THEN 3.00 * mk.sks
        WHEN n.grade = 'B-' THEN 2.70 * mk.sks
        WHEN n.grade = 'C+' THEN 2.30 * mk.sks
        WHEN n.grade = 'C' THEN 2.00 * mk.sks
```

```
        WHEN n.grade = 'D' THEN 1.00 * mk.sks
        WHEN n.grade = 'E' THEN 0.00 * mk.sks
        ELSE 0
    END AS 'hasil kali bobot dan sks'
FROM
    mahasiswa m
JOIN
    krs k ON m.id_mahasiswa = k.id_mahasiswa
JOIN
    krs_detail kd ON k.id_krs = kd.id_krs
JOIN
    nilai n ON kd.id_krs_detail = n.id_krs_detail
JOIN
    kelas kl ON kd.id_kelas = kl.id_kelas
JOIN
    mata_kuliah mk ON kl.id_mata_kuliah = mk.id_mata_kuliah
JOIN
    tahun_akademik ta ON k.id_tahun_akademik = ta.id_tahun_akademik
WHERE
    m.nim = @nim
    AND n.grade IS NOT NULL
    AND kd.status = 'Aktif'
    AND k.status = 'Disetujui'
ORDER BY
    ta.tahun, ta.semester, mk.kode_mata_kuliah;
SET @end := CURRENT_TIMESTAMP(6);


SELECT
    ROUND(TIME_TO_SEC(TIMEDIFF(@end, @start)) +
MICROSECOND(TIMEDIFF(@end, @start)) / 1000000) AS waktu_detik,
    TIME_TO_SEC(TIMEDIFF(@end, @start)) + MICROSECOND(TIMEDIFF(@end,
@start)) / 1000000 AS waktu_detik_float;
```

Penjelasan Query:

Skrip SQL ini digunakan untuk mengambil data nilai mahasiswa tertentu berdasarkan NIM (Nomor Induk Mahasiswa) dan menghitung waktu yang diperlukan untuk menjalankan query tersebut. Berikut penjelasan sederhana untuk setiap bagiannya:

1. Mencatat Waktu Mulai:

- `SET @start := CURRENT_TIMESTAMP(6);`

- Kode ini menyimpan waktu saat query dimulai ke dalam variabel `@start`, dengan presisi hingga mikrodetik. Ini digunakan untuk mengukur durasi eksekusi query.

2. Menyimpan NIM Mahasiswa:

- `SET @nim = 2019010340;`

- Variabel `@nim` diisi dengan NIM mahasiswa tertentu, dalam hal ini adalah `2019010340`. Query akan mencari data untuk mahasiswa ini.

3. Mengambil Data Nilai:

- Bagian ini mengambil data nilai mahasiswa dari berbagai tabel (seperti `mahasiswa`, `krs`, `nilai`, `mata_kuliah`, dll.) yang saling terhubung.

- Data yang diambil meliputi:

  - Nama mata kuliah (`nama_matkul`).

  - Jumlah SKS mata kuliah.

  - Nilai akhir mahasiswa dalam bentuk angka (`nilai_angka`) dan huruf (`nilai_huruf`).

  - Bobot nilai berdasarkan konversi nilai huruf (contoh: nilai A diberi bobot 4.00, B diberi bobot 3.00, dll.).

  - Hasil kali antara bobot nilai dan jumlah SKS mata kuliah.

- Data hanya diambil jika:

  - Mahasiswa tersebut memiliki nilai yang sudah dinilai (tidak NULL).

  - Status KRS detail adalah "Aktif".

  - Status KRS utama adalah "Disetujui".

- Data diurutkan berdasarkan tahun akademik, semester, dan kode mata kuliah.

4. Mencatat Waktu Selesai:

- `SET @end := CURRENT_TIMESTAMP(6);`

- Setelah query selesai, waktu ketika eksekusi selesai dicatat dalam variabel `@end`.

5. Menghitung Waktu Eksekusi:

- Bagian terakhir menghitung selisih waktu antara `@start` (waktu mulai) dan `@end` (waktu selesai) untuk mengetahui berapa lama query dijalankan.

- Hasilnya ditampilkan dalam dua format:

  - `waktu_detik`: Waktu dalam detik (dibulatkan).

  - `waktu_detik_float`: Waktu dalam detik dengan presisi lebih tinggi (tidak dibulatkan).

Table Output:

Percobaan Ke	waktu_detik	Waktu_dalam_detik	beda_raw
1	0,5255	1	0
2	0.5152	1	0
3	0.5594	1	0
4	0.4599	0	0
5	0.6153	1	0
6	8.1492	8	0
7	0.9683	1	0
8	1.0247	1	0
9	0.4443	0	0
10	0.7439	1	0
11	0.6682	1	0

3. Pengelompokan IPK Mahasiswa

Keluaran yang diharapkan:

angkatan, nim, prodi, ipk, kurang, sedang, tinggi.

Kategori:

Kurang: IPK < 2.75

Sedang: 2.75 ≤ IPK ≤ 3.2

Tinggi: IPK > 3.2

Jika ipk masuk ke kurang, maka kolom kurang akan terisi, begitu juga pada kolom sedang dan tinggi.

Ulangi 10x, catat waktu.

Query:

```
SET @start := CURRENT_TIMESTAMP(6);
SELECT
    m.tahun_masuk AS angkatan,
    m.nim,
    p.nama_prodi AS prodi,
    m.ipk,
    CASE WHEN m.ipk < 2.75 THEN 1 ELSE 0 END AS kurang,
    CASE WHEN m.ipk BETWEEN 2.75 AND 3.2 THEN 1 ELSE 0 END AS sedang,
    CASE WHEN m.ipk > 3.2 THEN 1 ELSE 0 END AS tinggi
FROM
    mahasiswa m
JOIN
    program_studi p ON m.id_prodi = p.id_prodi
WHERE
    m.status = 'Aktif'
ORDER BY
    m.tahun_masuk, p.nama_prodi, m.nim;
SET @end := CURRENT_TIMESTAMP(6);
SELECT
    ROUND(TIME_TO_SEC(TIMEDIFF(@end, @start)) + MICROSECOND(TIMEDIFF(@end,
@start)) / 1000000) AS waktu_detik,
    TIME_TO_SEC(TIMEDIFF(@end, @start)) + MICROSECOND(TIMEDIFF(@end,
```

```
@start)) / 1000000 AS waktu_detik_float;
```

Penjelasan Query:

Skrip SQL ini digunakan untuk mengambil data IPK mahasiswa aktif dari berbagai angkatan dan program studi, serta mengelompokkan IPK ke dalam kategori tertentu. Selain itu, skrip ini juga mengukur waktu yang dibutuhkan untuk menjalankan query.

1. **Mencatat Waktu Mulai Eksekusi**

```
SET @start := CURRENT_TIMESTAMP(6);
```

Menyimpan waktu mulai eksekusi query ke dalam variabel `@start`.

`CURRENT_TIMESTAMP(6)` mencatat waktu dengan presisi hingga mikrodetik (6 digit di belakang koma).

2. **Mengambil Data Mahasiswa dan Kategori IPK**

```
SELECT
    m.tahun_masuk AS angkatan,
    m.nim,
    p.nama_prodi AS prodi,
    m.ipk,
    CASE WHEN m.ipk < 2.75 THEN 1 ELSE 0 END AS kurang,
    CASE WHEN m.ipk BETWEEN 2.75 AND 3.2 THEN 1 ELSE 0 END AS sedang,
    CASE WHEN m.ipk > 3.2 THEN 1 ELSE 0 END AS tinggi
FROM
    mahasiswa m
JOIN
    program_studi p ON m.id_prodi = p.id_prodi
WHERE
    m.status = 'Aktif'
ORDER BY
    m.tahun_masuk, p.nama_prodi, m.nim;
```

Tujuan: Menampilkan daftar mahasiswa aktif beserta:

- Tahun masuk (angkatan)
- NIM
- Nama program studi
- IPK

Klasifikasi IPK:

- kurang: IPK < 2.75
- sedang: IPK antara 2.75 dan 3.20
- tinggi: IPK > 3.20

Masing-masing kategori diberi nilai 1 jika memenuhi syarat, 0 jika tidak.

Data diurutkan berdasarkan angkatan, nama prodi, dan NIM.

3. Mencatat Waktu Selesai Eksekusi

```
SET @end := CURRENT_TIMESTAMP(6);
```

Setelah pengambilan data selesai, waktu akhir dieksekusi dan disimpan dalam variabel `@end`

4. Menghitung Waktu Eksekusi Query

```
SELECT
    ROUND(TIME_TO_SEC(TIMEDIFF(@end, @start)) + MICROSECOND(TIMEDIFF(@end, @start)) / 1000000) AS waktu_detik,
    TIME_TO_SEC(TIMEDIFF(@end, @start)) + MICROSECOND(TIMEDIFF(@end, @start)) / 1000000 AS waktu_detik_float;
```

Fungsi yang digunakan:

- `TIMEDIFF(@end, @start)`: Menghitung selisih waktu antara mulai dan selesai.
- `TIME_TO_SEC(...)`: Mengubah selisih waktu menjadi detik.
- `MICROSECOND(...) / 1_000_000`: Mengubah mikrodetik menjadi pecahan detik**.**

Table Output:

Percobaan Ke	waktu_detik	Waktu_dalam_detik	beda_raw
1	4.2136	4	0
2	3.2924	3	0
3	30.2653	30	0
4	3.7134	4	0
5	31.2008	31	0
6	7.0651	7	0
7	3.1172	3	0

8	7.6712	8	0
9	4.8893	5	0
10	6.8886	7	0
11	4.2344	4	0

Output :

Percobaan 1

waktu_detik	waktu_detik_float
2	2,1186

Percobaan 2

waktu_detik	waktu_detik_float
1	1,0968

Percobaan 3

waktu_detik	waktu_detik_float
1	1,2639

Percobaan 4

waktu_detik	waktu_detik_float
3	2,8475

Percobaan 5

waktu_detik	waktu_detik_float
1	1,1807

Percobaan 6

waktu_detik	waktu_detik_float
1	1,3632

Percobaan 7

waktu_detik	waktu_detik_float
2	2,0534

Percobaan 8

waktu_detik	waktu_detik_float
2	1,7961

Percobaan 9

waktu_detik	waktu_detik_float
1	1,2513

Percobaan 10

waktu_detik	waktu_detik_float
1	0,9991

4. Analisis

Dari ketiga query yang dieksekusi masing-masing sebanyak 10 kali, berikut beberapa temuan umum yang dapat disimpulkan:

a. Query 1 (Menampilkan IPK Mahasiswa Angkatan 2019):
   i. Menggunakan join antar tabel mahasiswa dan program_studi, serta pemanggilan fungsi fn_hitung_ipk.
   ii. Eksekusi fungsi untuk setiap baris mahasiswa menyebabkan beban perhitungan meningkat.
   iii. Waktu eksekusi cenderung lebih lama karena adanya pemrosesan fungsi per record dan juga terpengaruh dengnan jaringan akses server.

b. Query 2 (Transkrip Mahasiswa per NIM):
   i. Melibatkan join kompleks ke lebih dari 6 tabel,eksekuis paling cepat.
   ii. Waktu eksekusi sangat bergantung pada indeks yang ada, terutama di tabel krs_detail, nilai, dan mata_kuliah.
   iii. Karena query difokuskan pada 1 NIM, performanya cukup stabil asal ada indeks di kolom yang digunakan pada WHERE.

c. Query 3 (Klasifikasi IPK Mahasiswa):=
   i. Performa cenderung menengah antara keduanya.
   ii. Tidak menggunakan fungsi agregat atau pemrosesan yang berat.
   iii. Kinerja ditentukan oleh jumlah data mahasiswa dan penggunaan indeks pada kolom status, id_prodi, dan ipk.

5. Kesimpulan dan saran optimasi

Query yang dijalankan telah berjalan dengan baik dan menghasilkan output yang sesuai dengan yang diharapkan. Data yang ditampilkan meliputi informasi mahasiswa aktif, lengkap dengan klasifikasi IPK berdasarkan interval yang telah ditentukan. Urutan data berdasarkan tahun masuk, program studi, dan NIM juga membantu dalam

pembacaan hasil.Dalam tabel juga kita sudah menggunakan INDEX agar mempercepat dalam pemrosesan query

Saran agar lebih maximal performa querynya ditambahkan indexnya sehingga query akan diproses lebih cepat,namun dipilih juga index yang di butuhkan jangan semuanya di berikan index karna bisa menurunkan performa,gunakan index secukupnya dan seperlunya saja,dan juga sering ada permasalahan route host unknow(mungin karna banyak yang akses ke servernya dan melakukan query bersama) jadi akan lebih baik jika di kerjakan dini hari atau ketika jam tidak melibatkan banyak pengguna