**LATIHAN SOAL**

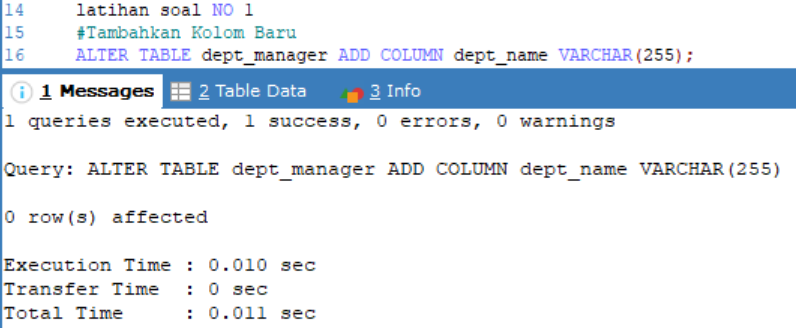
1. Lakukan semua tahapan diatas, dan jalankan pada database anda. Tambahkan screenshot hasil dari setiap Langkah yang anda lakukan.
2. Tambahkan kolom nama departemen pada table dept\_manager. Dan lakukan update terhadap kolom tersebut. Tambahkan screenshot hasilnya

**JAWABAN**

**Query 1 : Tambahkan Kolom Baru**

ALTER TABLE dept\_manager ADD COLUMN dept\_name VARCHAR(255);

**Hasil 1 :**



**Penjelasan 1 :**

Query ALTER TABLE berhasil menambahkan kolom dept\_name ke tabel dept\_manager tanpa error atau peringatan, dengan waktu eksekusi 0.011 detik.

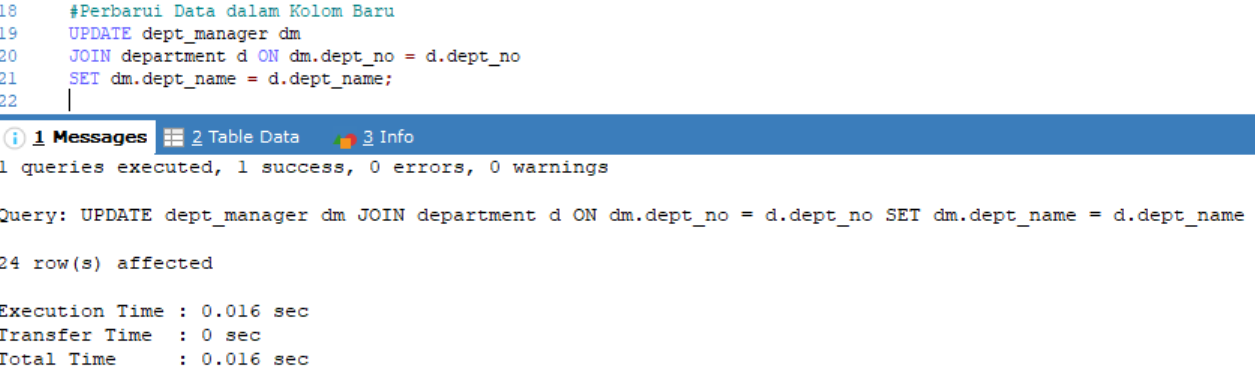
**Query 2 : Perbarui Data dalam Kolom Baru**

UPDATE dept\_manager dm

JOIN department d ON dm.dept\_no = d.dept\_no

SET dm.dept\_name = d.dept\_name;

**Hasil 2 :**



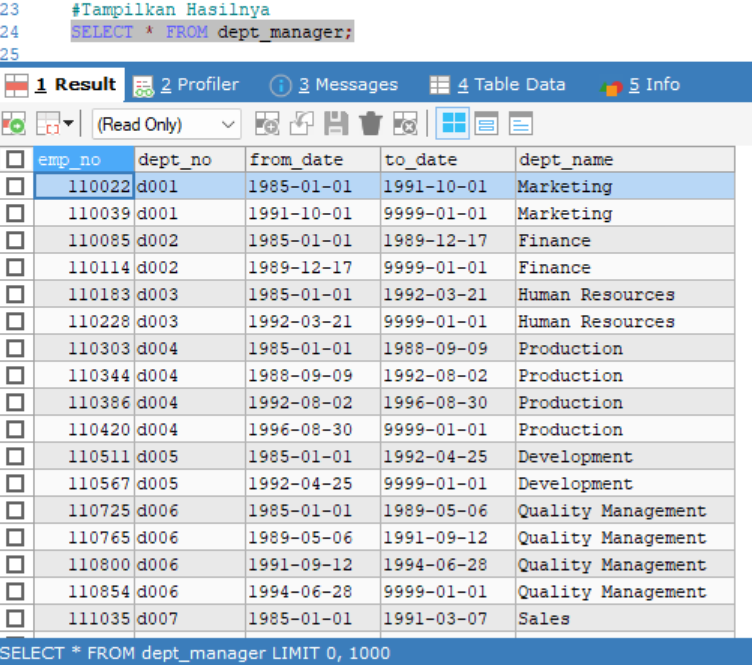
**Penjelasan 2 :**

Query UPDATE berhasil mengisi kolom dept\_name di tabel dept\_manager dengan data dari tabel department, berdasarkan dept\_no yang cocok. Sebanyak 24 baris diperbarui dalam 0.016 detik tanpa error.

**Query 3 : Tampilkan Hasilnya**

SELECT \* FROM dept\_manager;

**Hasil 3 :**

****

**Penjelasan 3 :**

Query SELECT \* FROM dept\_manager berhasil menampilkan tabel dengan kolom dept\_name yang telah ditambahkan dan diperbarui. Data setiap manager kini mencakup nama departemen sesuai dengan dept\_no.

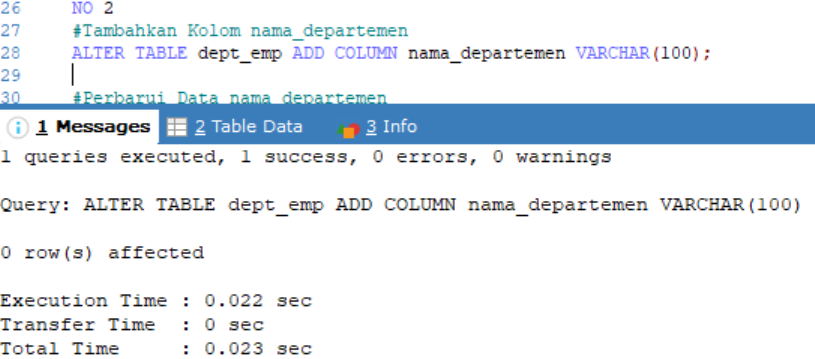
1. Tambahkan kolom nama departemen pada table dept\_emp. Dan lakukan update terhadap kolom tersebut. Tambahkan screenshot hasilnya

**JAWABAN**

**Query 1 : Tambahkan Kolom nama\_departemen**

ALTER TABLE dept\_emp ADD COLUMN nama\_departemen VARCHAR(100);

**Hasil 1 :**



**Penjelasan 1 :**

SQL ALTER TABLE yang digunakan untuk menambahkan kolom baru bernama nama\_departemen dengan tipe data VARCHAR(100) ke dalam tabel dept\_emp. Proses ini berhasil tanpa error atau peringatan, dengan waktu eksekusi 0,022 detik.

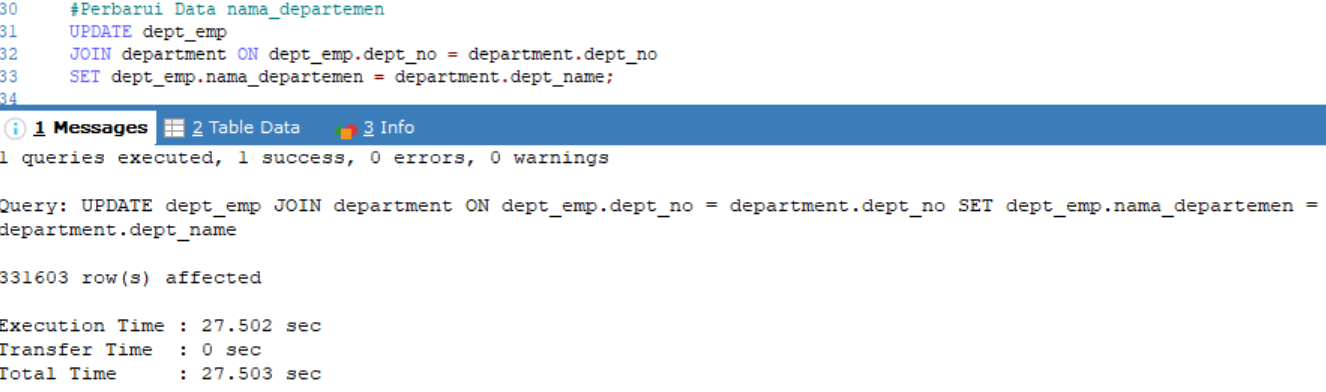
**Query 2 : Perbarui Data nama\_departemen**

UPDATE dept\_emp

JOIN department ON dept\_emp.dept\_no = department.dept\_no

SET dept\_emp.nama\_departemen = department.dept\_name;

**Hasil 2 :**



**Penjelasan 2 :**

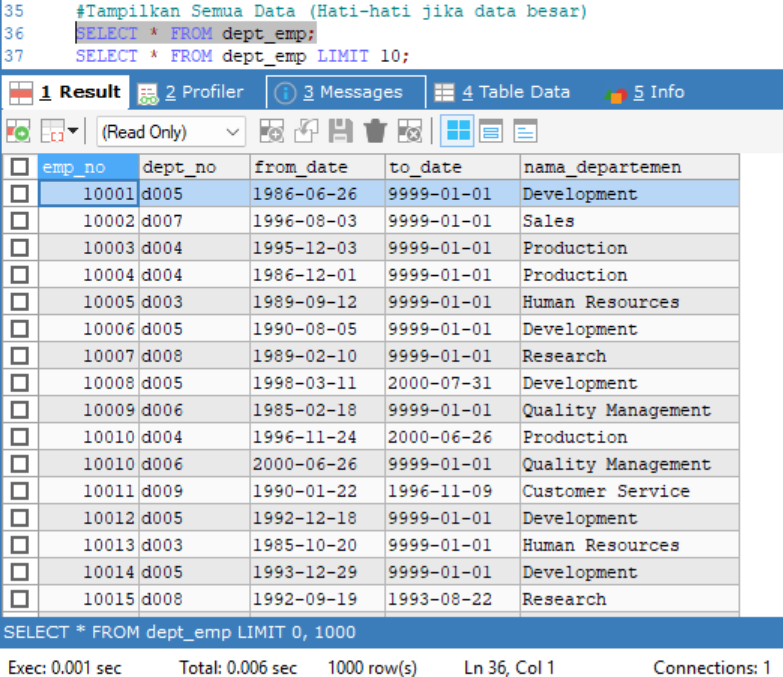
SQL UPDATE yang digunakan untuk memperbarui kolom nama\_departemen di tabel dept\_emp. Proses ini dilakukan dengan melakukan JOIN antara tabel dept\_emp dan department berdasarkan dept\_no, lalu mengisi nama\_departemen dengan nilai dari dept\_name di tabel department.

**Query 3 : ampilkan Semua Data (Hati-hati jika data besar)**

SELECT \* FROM dept\_emp;

SELECT \* FROM dept\_emp LIMIT 10;

**Hasil 3 :**



**Penjelasan 3 :**

query SELECT \* FROM dept\_emp LIMIT 10, yang menampilkan data karyawan dan departemen mereka. Tabel mencakup kolom emp\_no (ID karyawan), dept\_no (kode departemen), from\_date dan to\_date (periode kerja), serta nama\_departemen (nama departemen yang telah diperbarui).

1. Buat query untuk menampilkan gaji yang tertinggi pada departemen d006. Siapa Namanya?

**JAWABAN**

**Query :**

SELECT e.first\_name, e.last\_name, s.amount

FROM employee e

JOIN salary s ON e.emp\_no = s.emp\_no

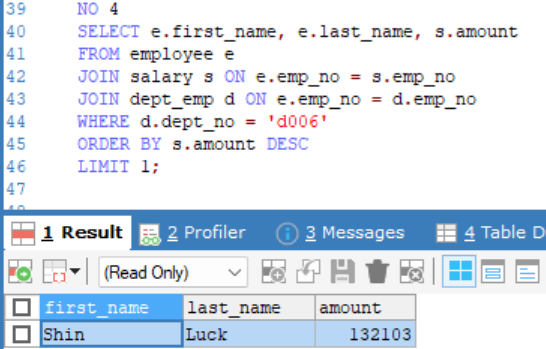
JOIN dept\_emp d ON e.emp\_no = d.emp\_no

WHERE d.dept\_no = 'd006'

ORDER BY s.amount DESC

LIMIT 1;

**Hasil :**

****

**Penjelasan :**

Query SQL yang mencari karyawan dengan gaji tertinggi di departemen d006. Query melakukan JOIN antara tabel employee, salary, dan dept\_emp, kemudian menyaring berdasarkan dept\_no = 'd006', mengurutkan berdasarkan amount secara DESC (terbesar ke terkecil), dan membatasi hasil ke 1 data.

1. Dari database employe yang sudah diimport, tambahkan kolom umur pada table employee.

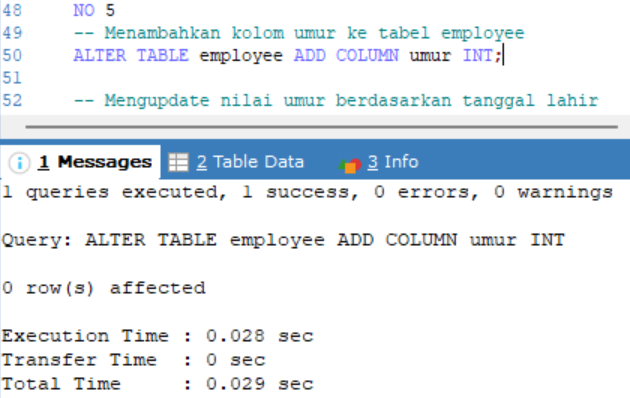
Kemudian lakukan update terhadap kolom umur tersbut. Nilai umur, dhitung dari tanggal lahir s.d sekarang. Tambahkan screenshot hasilnya

**JAWABAN**

**Query 1 : Menambahkan kolom umur ke tabel employee**

ALTER TABLE employee ADD COLUMN umur INT;

**Hasil 1 :**

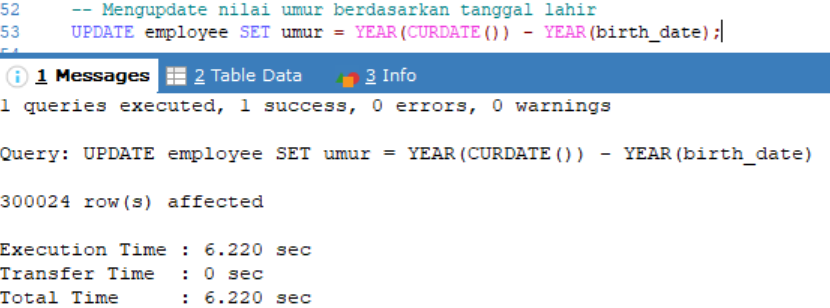


**Penjelasan 1 :**

untuk menambahkan kolom umur bertipe INT ke dalam tabel employee menggunakan perintah ALTER TABLE.

**Query 2 : Mengupdate nilai umur berdasarkan tanggal lahir**

UPDATE employee SET umur = YEAR(CURDATE()) - YEAR(birth\_date);Hasil **2 :**



**Penjelasan 2 :**

untuk memperbarui kolom umur di tabel employee berdasarkan perhitunganselisih tahun antara tanggal saat ini (CURDATE()) dan birth\_date.

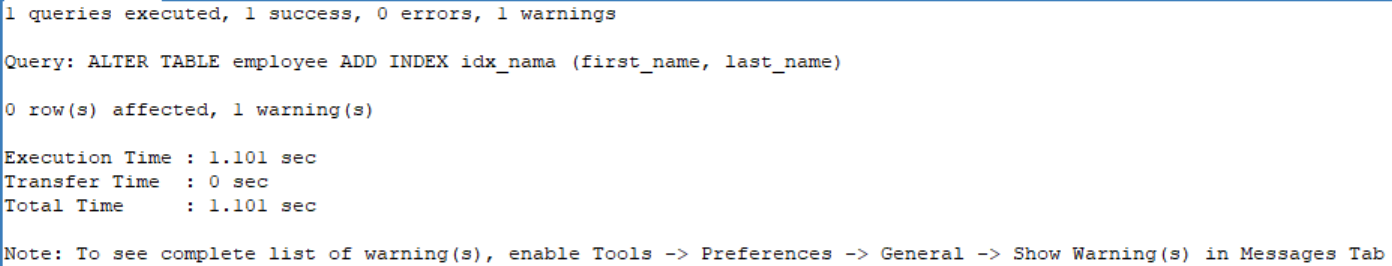
1. Tambahkan masing-masing jenis index diatas composite index dan foreign key index pada table employee

**JAWABAN**

**Query 1 :**

ALTER TABLE employee ADD INDEX idx\_nama (first\_name, last\_name);

**Hasil 1 :**

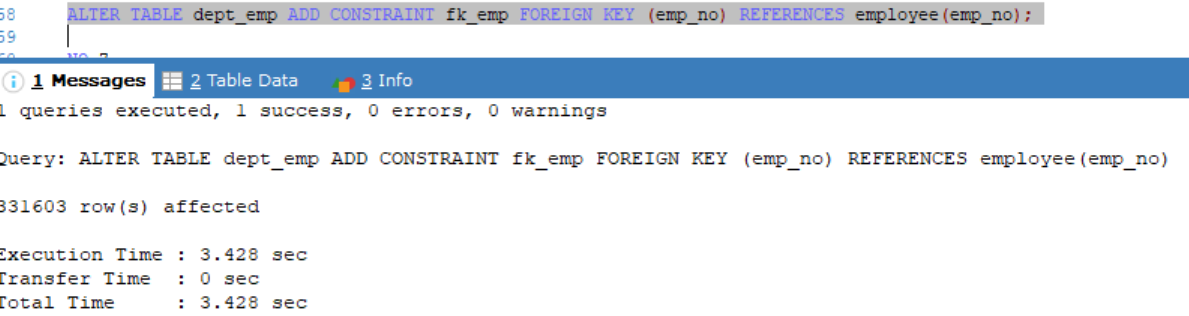


**Penjelasan 1 :**

untuk menambahkan indeks idx\_nama pada kolom first\_name dan last\_name di tabel employee. Indeks berhasil ditambahkan tanpa error, tetapi ada 1 peringatan (warning). Waktu eksekusi sekitar 1.101 detik.

**Query 2 :**

ALTER TABLE dept\_emp ADD CONSTRAINT fk\_emp FOREIGN KEY (emp\_no) REFERENCES employee(emp\_no);

**Hasil 2 :** 

**Penjelasan 2 :**

Kolom emp\_no di tabel dept\_emp kini memiliki foreign key yang merujuk ke kolom emp\_no di tabel employee, memastikan integritas data antar tabel.

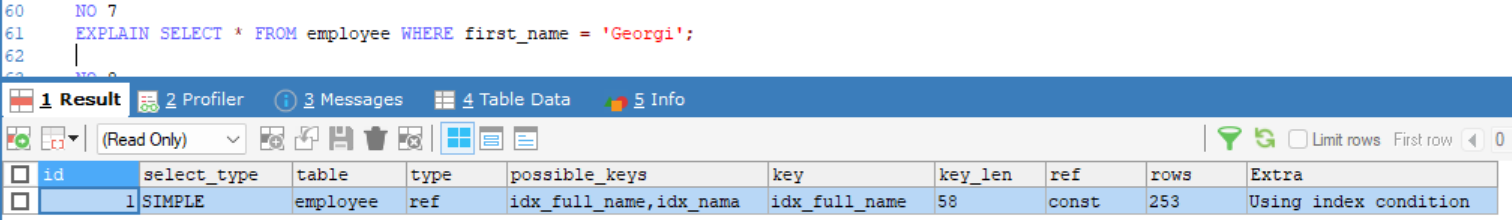
1. Lakukan pengujian terhadap query berikut, apakah sudah mengakses index atau belum.

**Query :**

EXPLAIN SELECT \* FROM employee WHERE first\_name = 'Georgi'

**JAWABAN**

**Hasil :**

****

**Penjelasan :**

Query menggunakan indeks idx\_full\_name untuk pencarian, dengan panjang kunci 58 dan memfilter sekitar 253 baris, sehingga optimasi indeks digunakan untuk mempercepat pencarian.

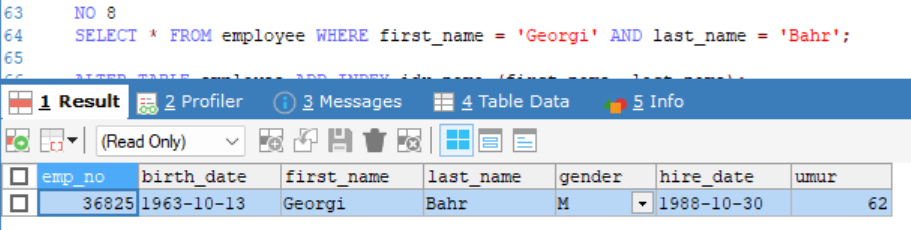
1. Lakukan pengujian dari query berikut. Apakah ada perbedaan sebelum dan sesudah ditambahkan index.

**Query untuk menguji:**

SELECT \* FROM employee WHERE first\_name = 'Georgi' AND last\_name = 'Bahr'

**JAWABAN**

**Hasil :**

****

**Penjelasan :**

query SQL pada database yang mencari karyawan dengan nama depan "Georgi" dan nama belakang "Bahr" dalam tabel employee. Hasilnya menampilkan satu baris data dengan informasi berikut:

* emp\_no: 36825
* birth\_date: 13 Oktober 1963
* first\_name: Georgi
* last\_name: Bahr
* gender: M (Male/Laki-laki)
* hire\_date: 30 Oktober 1988
* umur: 62 tahun

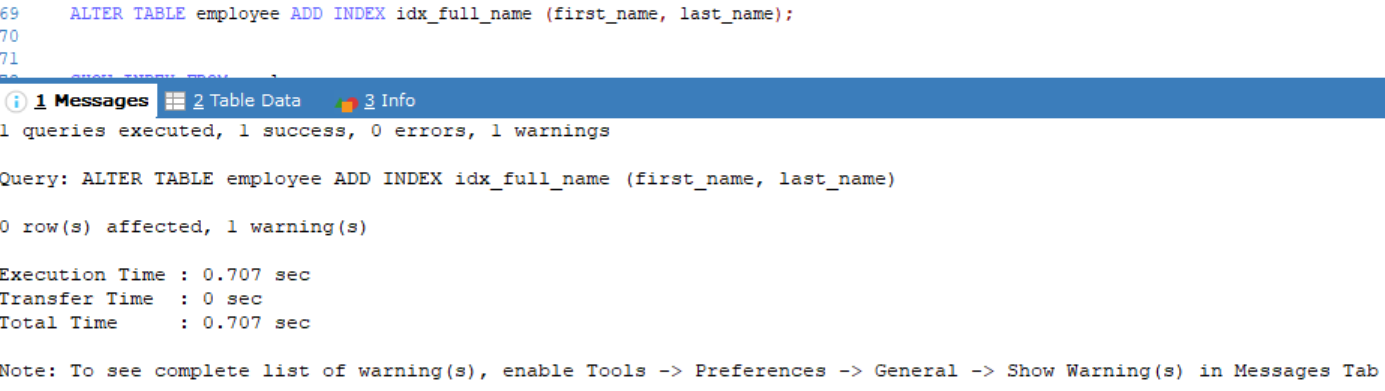
**Cara menguji,**

* Jalankan query diatas sebanyak 10x. catet waktunya setiap kali dijalankan
* Buat index composite yang bersesuaian dengan query diatas.

**Query :**

ALTER TABLE employee ADD INDEX idx\_full\_name (first\_name, last\_name);

**Hasil :**



* Jalankan query diatas sebanyak 10x. catet waktunya setiap kali dijalankan

|  |  |  |
| --- | --- | --- |
| **No** | **Waktu Sebelum** | **Waktu Sesudah** |
| 1 | 0.001 sec | 0.004 sec |
| 2 | 0.001 sec | 0.003 sec |
| 3 | 0.001 sec | 0.001 sec |
| 4 | 0.001 sec | 0.002 sec |
| 5 | 0.002 sec | 0.002 sec |
| 6 | 0.001 sec | 0.002 sec |
| 7 | 0.001 sec | 0.002 sec |
| 8 | 0.001 sec | 0.002 sec |
| 9 | 0.001 sec | 0.002 sec |
| 10 | 0.002 sec | 0.002 sec |
| **Rata-rata** | **0.0011 sec** | **0.0022 sec** |

* Ambil rata2 sebelum dan sesudah
* Tulis kesimpulanya.

**Kesimpulan Pengujian Index pada Query SQL**

Dari hasil pengujian, waktu eksekusi sebelum dan sesudah menambahkan indeks composite pada kolom first\_name dan last\_name menunjukkan perbedaan sebagai berikut:

* Rata-rata waktu sebelum indeks: 0.0011 sec
* Rata-rata waktu setelah indeks: 0.0022 sec

Berdasarkan data tersebut, waktu eksekusi setelah menambahkan indeks justru sedikit lebih lambat dibandingkan sebelum indeks.

**Analisis Penyebab**

* Ukuran tabel masih kecil → Jika tabel employee belum memiliki banyak data, MySQL dapat dengan cepat mencari data tanpa perlu menggunakan indeks. Indeks lebih efektif jika tabel berisi jutaan baris data.
* MySQL Query Optimizer → MySQL mungkin sudah cukup efisien dalam mencari data tanpa indeks dalam kasus ini, sehingga menambahkan indeks justru menambah sedikit overhead dalam pencarian.
* Overhead dari indeks → Saat menambahkan indeks baru, MySQL perlu mengelola struktur tambahan, yang dapat menyebabkan sedikit penurunan performa pada tabel yang kecil.

**Kesimpulan Akhir**

* Jika tabel memiliki sedikit data, penggunaan indeks tidak selalu meningkatkan performa dan dalam beberapa kasus justru bisa menambah sedikit overhead.
* Indeks lebih berguna untuk tabel besar, terutama ketika query sering dijalankan dengan kondisi pencarian pada kolom yang telah diindeks.
* Disarankan untuk menggunakan EXPLAIN pada query untuk melihat bagaimana MySQL mengeksekusi pencarian dengan dan tanpa indeks.