

# **LAPORAN PRAKTIKUM BASIS DATA**

## **KELOMPOK 2**



### **DISUSUN OLEH :**

Oktario Mufti Yudha (2320506044)

Azqia Fattimah A R (2340506067)

Siwi Kartika Dewi (2340506075)

**PROGRAM STUDI S1 TEKNOLOGI INFORMASI  
JURUSAN TEKNIK ELEKTRO FAKULTAS TEKNIK  
UNIVERSITAS TIDAR**

**2024**

## **I. Dasar Teori**

Database atau basis data adalah kumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling berhubungan sehingga mudah dalam pengelolaannya. Melalui pengelolaan tersebut pengguna dapat memperoleh kemudahan dalam mencari informasi, menyimpan informasi dan membuang informasi.

Pengelolaan record dalam basis data adalah proses yang melibatkan penyimpanan, pengambilan, pembaruan, dan penghapusan data dalam sebuah sistem basis data. Pengelolaan ini sangat penting untuk memastikan integritas, keamanan, dan efisiensi akses data. Berbagai teknik dan alat digunakan untuk mengelola record, termasuk indeks, kunci utama, dan kunci asing, yang semuanya membantu mempercepat pencarian dan memastikan hubungan antar tabel tetap konsisten.

SQL Subquery, atau subquery SQL, adalah sebuah query di dalam query lain yang digunakan untuk mengambil data yang kemudian digunakan oleh query utama. Subquery dapat berada dalam klausa SELECT, FROM, atau WHERE, dan berfungsi untuk memecah tugas yang kompleks menjadi bagian-bagian yang lebih sederhana. Misalnya, subquery dalam klausa WHERE dapat digunakan untuk membandingkan nilai terhadap hasil dari query lain, memberikan fleksibilitas dan kekuatan dalam mengambil data. Penggunaan subquery sering kali membuat query lebih mudah dibaca dan dipelihara.

SQL Join adalah operasi dalam SQL yang digunakan untuk menggabungkan record dari dua atau lebih tabel berdasarkan suatu kondisi yang terkait. Ada beberapa jenis join, termasuk INNER JOIN, LEFT JOIN, RIGHT JOIN, dan FULL OUTER JOIN. INNER JOIN hanya mengembalikan record yang memiliki nilai yang cocok di kedua tabel. LEFT JOIN mengembalikan semua record dari tabel kiri dan record yang cocok dari tabel kanan, atau NULL jika tidak ada kecocokan. SQL Join sangat berguna untuk menggabungkan data yang tersebar di beberapa tabel dan menghasilkan set data yang komprehensif untuk analisis lebih lanjut.

Prosedur dan fungsi adalah blok kode yang dapat digunakan kembali dan diatur dalam sistem basis data untuk melakukan tugas-tugas tertentu. Prosedur (stored procedures) adalah sekumpulan perintah SQL yang disimpan dan dijalankan di server basis data, yang dapat menerima input, menjalankan logika, dan mengembalikan hasil. Fungsi (functions) mirip dengan prosedur tetapi biasanya mengembalikan nilai tunggal dan dapat digunakan dalam pernyataan SQL seperti SELECT dan WHERE. Keduanya

membantu mengotomatisasi tugas rutin, meningkatkan efisiensi, dan menjaga konsistensi dalam pengolahan data.

Trigger adalah sejenis prosedur yang dijalankan secara otomatis sebagai respons terhadap peristiwa tertentu dalam tabel basis data, seperti insert, update, atau delete. Trigger digunakan untuk memastikan integritas data, menegakkan aturan bisnis, dan mencatat perubahan dalam tabel audit. Misalnya, sebuah trigger dapat digunakan untuk memperbarui nilai di tabel lain setiap kali terjadi perubahan pada tabel tertentu, atau untuk mencegah operasi yang melanggar aturan integritas referensial. Penggunaan trigger membantu menjaga basis data tetap konsisten dan aman tanpa intervensi manual.

## **II. Metode Praktikum**

### **A. Alat dan bahan**

Alat :

1. PC (Komputer)
2. Keyboard
3. Mouse

Bahan :

1. Operating System Windows 10
2. Aplikasi Paket Web server XAMPP
3. phpMyAdmin
4. File Materi Praktikum

### **B. Langkah kerja**

1. Menyalakan perangkat komputer
2. Membuka file materi praktikum yang di kirimkan dosen
3. Mengerjakan praktikum dan tugas yang ada pada file materi praktikum

### III. Hasil dan Analisis

Membuat beberapa tabel berdasarkan ERD yang telah dibuat sebelumnya. Yang pertama yaitu membuat tabel karyawan yang berisi id\_karyawan tipe data int(11) berperan sebagai primary key, nama\_karyawan tipe datanya varchar(100), dan alamat tipe datanya varchar(255).

#	Nama	Jenis
<input type="checkbox"/> 1	id_karyawan	int(11)
<input type="checkbox"/> 2	nama_karyawan	varchar(100)
<input type="checkbox"/> 3	alamat	varchar(255)

*Tabel karyawan*

Tabel kedua yaitu tabel pekerjaan, yang berisi kode\_pekerjaan tipe datanya int(11) berperan sebagai primary key dan nama\_pekerjaan tipe datanya varchar(100).

#	Nama	Jenis
<input type="checkbox"/> 1	kode_pekerjaan	int(11)
<input type="checkbox"/> 2	nama_pekerjaan	varchar(100)

*Tabel pekerjaan*

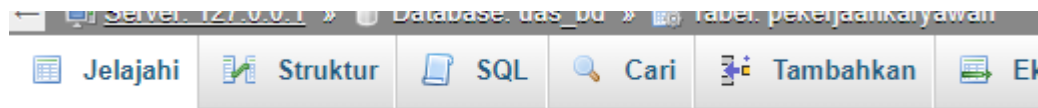
Tabel ketiga yaitu pekerjaankaryawan, yaitu berisi id\_karyawan tipe datanya int(11) berperan sebagai primary key dan kode\_pekerjaan tipe datanya int(11) berperan sebagai primary juga. Adanya dua primary key di tabel tersebut menunjukkan penggunaan primary key gabungan untuk memastikan keunikan kombinasi id\_karyawan dan kode\_pekerjaan. Hal ini sering digunakan dalam tabel yang mengelola hubungan banyak-ke-banyak atau dalam situasi di mana satu kolom saja tidak cukup untuk memastikan keunikan data.

#	Nama	Jenis
<input type="checkbox"/> 1	id_karyawan	int(11)
<input type="checkbox"/> 2	kode_pekerjaan	int(11)

*Tabel pekerjaankaryawan*

## 1. Record dalam basis data

Memasukkan beberapa data pada setiap tabel. Caranya yaitu mengklik tabel yang akan di tambahkan data dan klik ‘tambahkan’ pada menu di bagian atas tersebut.



Lalu tampilannya akan seperti gambar di bawah ini, di sini kolom-kolom yang tersedia dengan data yang di inginkan, setelah selesai klik tombol ‘irim’ untuk menyimpan data baru ke dalam tabel, kemudian ulangi langkah-langkah tersebut untuk memasukkan data pada tabel yang lain.

Kolom	Jenis	Fungsi	Tak Ternilai	Nilai
id_karyawan	int(11)	<input type="text"/>	<input type="text"/>	<input type="text"/>
kode_pekerjaan	int(11)	<input type="text"/>	<input type="text"/>	<input type="text"/>

`SELECT * FROM `karyawan``

☐ Profil [\[ Edit dikotak \]](#) [\[ Ubah \]](#) [\[ Jelaskan SQL \]](#) [\[ Buat kode PHP \]](#) [\[ Segarkan \]](#)

☐ Tampilkan semua | Jumlah baris:  | Saring baris:

Extra options

		id_karyawan	nama_karyawan	alamat
<input type="checkbox"/>	Ubah	Salin	Hapus	1 Siwi Magelang
<input type="checkbox"/>	Ubah	Salin	Hapus	2 Azkia Magelang
<input type="checkbox"/>	Ubah	Salin	Hapus	3 Oktario Purworejo
<input type="checkbox"/>	Ubah	Salin	Hapus	4 Kartika Magelang

`SELECT * FROM `pekerjaan``

☐ Profil [\[ Edit dikotak \]](#) [\[ Ubah \]](#) [\[ Jelaskan SQL \]](#) [\[ Buat](#)

☐ Tampilkan semua | Jumlah baris:

Extra options

<div>←T→</div>				kode_pekerjaan	nama_pekerjaan
<input type="checkbox"/>		Ubah	Salin  Hapus	1	Mahasiswa
<input type="checkbox"/>		Ubah	Salin  Hapus	2	Bos Muda
<input type="checkbox"/>		Ubah	Salin  Hapus	3	Analyst

```
SELECT * FROM `pekerjaankaryawan`
```

☐ Profil [ Edit dikotak ] [ Ubah ] [ Jelaskan SQL ]

☐ Tampilkan semua | Jumlah baris: 25 ▼

Extra options

			id_karyawan	kode_pekerjaan
<input type="checkbox"/>	Ubah	Salin	Hapus	1 3
<input type="checkbox"/>	Ubah	Salin	Hapus	4 1

## 2. SQL Subquery

### A. Subquery SELECT

Subquery dalam pernyataan SELECT digunakan untuk menghasilkan nilai tambahan untuk setiap baris yang dihasilkan oleh query utama. Subquery ini dieksekusi sekali untuk setiap baris yang ditemukan dalam hasil query utama. Pada subquery (`SELECT COUNT(*) FROM pekerjaankaryawan WHERE pekerjaankaryawan.id_karyawan = karyawan.id_karyawan`) menghitung jumlah pekerjaan yang dimiliki oleh setiap karyawan. Hasil subquery ini akan dimasukkan sebagai kolom tambahan dengan nama `jumlah_pekerjaan` dalam hasil query utama.

```
SELECT nama_karyawan, (SELECT COUNT(*) FROM pekerjaankaryawan WHERE
pekerjaankaryawan.id_karyawan = karyawan.id_karyawan) AS jumlah_pekerjaan FROM
karyawan;
```

☐ Profil [ Edit dikotak ] [ Ubah ] [ Jelaskan SQL ] [ Buat kode PHP ] [

☐ Tampilkan semua | Jumlah baris: 25 ▼

Extra options

nama_karyawan	jumlah_pekerjaan
Siwi	1
Azkie	0
Oktario	0
Kartika	1

### B. Subquery WHERE

Subquery dalam pernyataan WHERE digunakan untuk membatasi hasil query utama dengan nilai yang dihasilkan oleh subquery tersebut. Subquery WHERE dieksekusi terlebih dahulu dan hasilnya digunakan untuk memfilter baris-baris dalam tabel yang dievaluasi oleh query utama. Pada

subquery (SELECT id\_karyawan FROM pekerjaankaryawan WHERE kode\_pekerjaan = '123') menghasilkan daftar id karyawan yang memiliki pekerjaan dengan kode '123'. Kemudian, query utama menggunakan daftar ini untuk memfilter baris-baris dalam tabel karyawan sehingga hanya baris-baris yang memiliki id karyawan yang sesuai yang akan dimasukkan dalam hasil query utama.

```
SELECT * FROM karyawan WHERE id_karyawan IN (SELECT id_karyawan FROM pekerjaankaryawan WHERE kode_pekerjaan = '1');
```

☐ Profil [\[ Edit dikotak \]](#) [\[ Ubah \]](#) [\[ Jelaskan SQL \]](#) [\[ Buat kode PHP \]](#) [\[ Segarkan \]](#)

☐ Tampilkan semua | Jumlah baris: 25

Extra options

	id_karyawan	nama_karyawan	alamat
<input type="checkbox"/> <a href="#">Ubah</a> <a href="#">Salin</a> <a href="#">Hapus</a>	4	Kartika	Magelang

### 3. SQL JOIN

#### A. INNER JOIN

Mengembalikan baris yang memiliki nilai yang cocok dalam kedua tabel yang bergabung. INNER JOIN akan menghasilkan set gabungan dari kedua tabel, di mana baris yang tidak memiliki padanan dalam kedua tabel tersebut akan diabaikan.

```
SELECT karyawan.id_karyawan, karyawan.nama_karyawan, karyawan.alamat, pekerjaan.kode_pekerjaan, pekerjaan.nama_pekerjaan FROM karyawan INNER JOIN pekerjaankaryawan ON karyawan.id_karyawan = pekerjaankaryawan.id_karyawan INNER JOIN pekerjaan ON pekerjaankaryawan.kode_pekerjaan = pekerjaan.kode_pekerjaan;
```

☐ Profil [\[ Edit dikotak \]](#) [\[ Ubah \]](#) [\[ Jelaskan SQL \]](#) [\[ Buat kode PHP \]](#) [\[ Segarkan \]](#)

☐ Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini | Sort by key: Tidak ada

Extra options

id_karyawan	nama_karyawan	alamat	kode_pekerjaan	nama_pekerjaan
4	Kartika	Magelang	1	Mahasiswa
1	Siwi	Magelang	3	Analyst

#### B. LEFT JOIN

Mengembalikan semua baris dari tabel di sisi kiri (tabel pertama yang disebutkan dalam pernyataan JOIN), dan baris dari tabel di sisi kanan (tabel kedua yang disebutkan dalam pernyataan JOIN) yang memiliki nilai yang cocok dalam kolom yang dijadikan acuan. Jika tidak ada nilai yang cocok di tabel di sisi kanan, nilai-nilai untuk kolom di tabel di sisi kanan akan berisi NULL.

```
SELECT karyawan.id_karyawan, karyawan.nama_karyawan, karyawan.alamat,
pekerjaan.kode_pekerjaan, pekerjaan.nama_pekerjaan FROM karyawan LEFT JOIN
pekerjaankaryawan ON karyawan.id_karyawan = pekerjaankaryawan.id_karyawan LEFT JOIN
pekerjaan ON pekerjaankaryawan.kode_pekerjaan = pekerjaan.kode_pekerjaan;
```

☐ Profil [ Edit dikotak ] [ Ubah ] [ Jelaskan SQL ] [ Buat kode PHP ] [ Segarkan ]

☐ Tampilkan semua | Jumlah baris: 25 ▼

Extra options

id_karyawan	nama_karyawan	alamat	kode_pekerjaan	nama_pekerjaan
1	Siwi	Magelang	3	Analyst
2	Azkia	Magelang	NULL	NULL
3	Oktario	Purworejo	NULL	NULL
4	Kartika	Magelang	1	Mahasiswa

### C. RIGHT JOIN

Kebalikan dari LEFT JOIN. Mengembalikan semua baris dari tabel di sisi kanan (tabel kedua yang disebutkan dalam pernyataan JOIN), dan baris dari tabel di sisi kiri (tabel pertama yang disebutkan dalam pernyataan JOIN) yang memiliki nilai yang cocok dalam kolom yang dijadikan acuan. Jika tidak ada nilai yang cocok di tabel di sisi kiri, nilai-nilai untuk kolom di tabel di sisi kiri akan berisi NULL.

```
SELECT karyawan.id_karyawan, karyawan.nama_karyawan, karyawan.alamat,
pekerjaan.kode_pekerjaan, pekerjaan.nama_pekerjaan FROM karyawan RIGHT JOIN
pekerjaankaryawan ON karyawan.id_karyawan = pekerjaankaryawan.id_karyawan RIGHT JOIN
pekerjaan ON pekerjaankaryawan.kode_pekerjaan = pekerjaan.kode_pekerjaan;
```

☐ Profil [ Edit dikotak ] [ Ubah ] [ Jelaskan SQL ] [ Buat kode PHP ] [ Segarkan ]

☐ Tampilkan semua | Jumlah baris: 25 ▼

Extra options

id_karyawan	nama_karyawan	alamat	kode_pekerjaan	nama_pekerjaan
4	Kartika	Magelang	1	Mahasiswa
NULL	NULL	NULL	2	Bos Muda
1	Siwi	Magelang	3	Analyst

### D. FULL JOIN

Mengembalikan semua baris dari kedua tabel yang bergabung, dengan nilai-nilai yang cocok dari setiap tabel ketika tersedia. Jika tidak ada nilai yang cocok, nilai-nilai untuk kolom dari tabel yang tidak memiliki padanan akan berisi NULL.



```

SELECT karyawan.id_karyawan, karyawan.nama_karyawan, karyawan.alamat,
pekerjaan.kode_pekerjaan, pekerjaan.nama_pekerjaan FROM karyawan LEFT JOIN
pekerjaankaryawan ON karyawan.id_karyawan = pekerjaankaryawan.id_karyawan LEFT JOIN
pekerjaan ON pekerjaankaryawan.kode_pekerjaan = pekerjaan.kode_pekerjaan UNION
SELECT karyawan.id_karyawan, karyawan.nama_karyawan, karyawan.alamat,
pekerjaan.kode_pekerjaan, pekerjaan.nama_pekerjaan FROM karyawan RIGHT JOIN

```

☐ Profil [ Edit dikotak ] [ Ubah ] [ Jelaskan SQL ] [ Buat kode PHP ] [ Segarkan ]

☐ Tampilkan semua | Jumlah baris: 25 ▼

Extra options

id_karyawan	nama_karyawan	alamat	kode_pekerjaan	nama_pekerjaan
1	Siwi	Magelang	3	Analyst
2	Azkie	Magelang	NULL	NULL
3	Oktario	Purworejo	NULL	NULL
4	Kartika	Magelang	1	Mahasiswa
NULL	NULL	NULL	2	Bos Muda

#### 4. Prosedur dan Fungsi

##### A. Prosedur tambah karyawan

Prosedur ini merupakan stored procedure dalam SQL yang bertujuan untuk memasukkan data baru ke dalam dua tabel, yaitu Karyawan dan PekerjaanKaryawan. Pertama, data karyawan seperti id, nama, dan alamat dimasukkan ke dalam tabel Karyawan, diikuti dengan memasukkan informasi pekerjaan karyawan ke dalam tabel PekerjaanKaryawan. Nilai-nilai yang dimasukkan berasal dari parameter yang dilewatkan ke dalam prosedur, sehingga prosedur ini memungkinkan untuk memasukkan data dengan nilai yang dinamis sesuai dengan argumen yang diberikan ketika memanggil prosedur.

Ubah

Nama routine

TambahKaryawan

Jenis

PROCEDURE ▼

Parameter

Arah	Nama	Jenis	Panjang/Nilai
↑ IN ▼	p_id_karyawan	INT ▼	
↑ IN ▼	p_nama_karyawan	VARCHAR ▼	100
↑ IN ▼	p_alamat	VARCHAR ▼	255
↑ IN ▼	p_kode_pekerjaan	INT ▼	

Tambahkan parameter

```

1 BEGIN
2     INSERT INTO Karyawan (id_karyawan, nama_karyawan, alamat)
3     VALUES (p_id_karyawan, p_nama_karyawan, p_alamat);
4
5     INSERT INTO PekerjaanKaryawan (id_karyawan, kode_pekerjaan)
6     VALUES (p_id_karyawan, p_kode_pekerjaan);
7 END

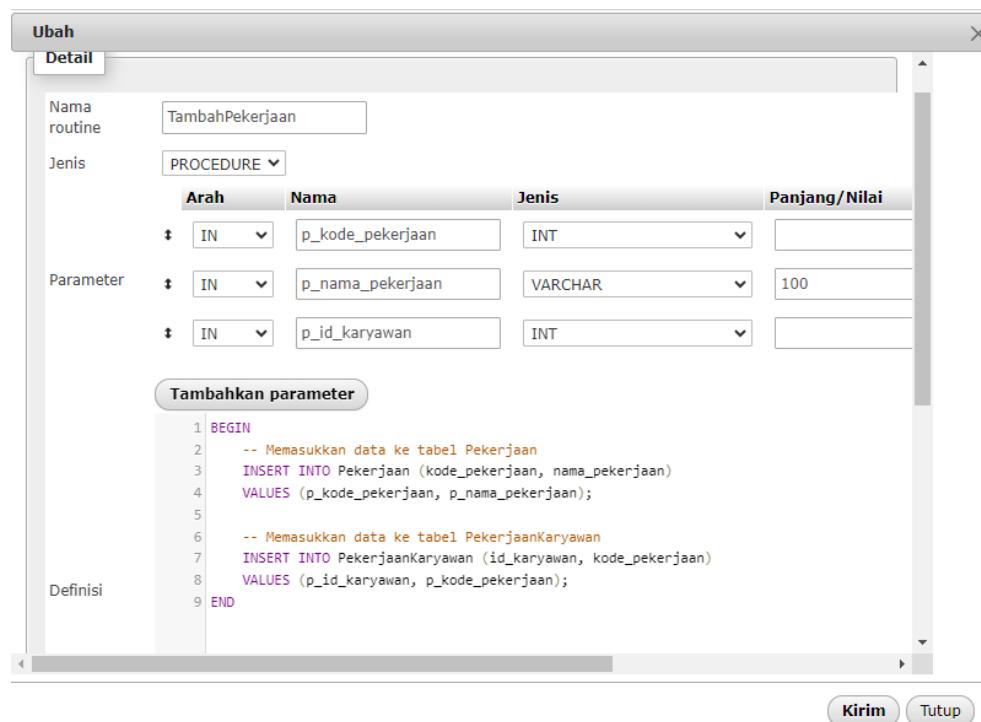
```

Kirim

Tutup

## B. Prosedur tambah pekerjaan

Prosedur ini merupakan stored procedure dalam SQL yang bertujuan untuk memasukkan data baru ke dalam tabel Pekerjaan dan PekerjaanKaryawan. Pertama, data pekerjaan seperti kode pekerjaan dan nama pekerjaan dimasukkan ke dalam tabel Pekerjaan. Kemudian, data tersebut juga dihubungkan dengan karyawan dengan memasukkan informasi ke dalam tabel PekerjaanKaryawan. Nilai-nilai yang dimasukkan berasal dari parameter yang dilewatkan ke dalam prosedur, sehingga prosedur ini memungkinkan untuk memasukkan data dengan nilai yang dinamis sesuai dengan argumen yang diberikan ketika memanggil prosedur.



**Ubah**

**Detail**

Nama routine: TambahPekerjaan

Jenis: PROCEDURE

	Arah	Nama	Jenis	Panjang/Nilai
Parameter	IN	p_kode_pekerjaan	INT	
	IN	p_nama_pekerjaan	VARCHAR	100
	IN	p_id_karyawan	INT	

**Tambahkan parameter**

**Definisi**

```
1 BEGIN
2 -- Memasukkan data ke tabel Pekerjaan
3 INSERT INTO Pekerjaan (kode_pekerjaan, nama_pekerjaan)
4 VALUES (p_kode_pekerjaan, p_nama_pekerjaan);
5
6 -- Memasukkan data ke tabel PekerjaanKaryawan
7 INSERT INTO PekerjaanKaryawan (id_karyawan, kode_pekerjaan)
8 VALUES (p_id_karyawan, p_kode_pekerjaan);
9 END
```

**Kirim** **Tutup**

## 5. Trigger

### A. Trigger input karyawan

Trigger ini adalah sebuah trigger AFTER INSERT dalam SQL. Trigger yang dirancang untuk otomatis memasukkan data baru ke dalam tabel PekerjaanKaryawan setiap kali operasi INSERT terjadi pada tabel yang memicu trigger ini. Nilai id\_karyawan dari baris yang baru saja dimasukkan ke dalam tabel induk diambil menggunakan variabel khusus NEW.id\_karyawan, dan nilai 1 digunakan sebagai nilai statis untuk kode\_pekerjaan. Dengan demikian, trigger ini memungkinkan untuk mengaitkan karyawan baru dengan pekerjaan yang telah ditentukan secara otomatis setelah ditambahkan ke dalam sistem.

Ubah

Detail

Nama trigger

after\_karyawan\_insert

Tabel

karyawan

Waktu

AFTER

Kejadian

INSERT

```

1 BEGIN
2   INSERT INTO PekerjaanKaryawan (id_karyawan,
3     kode_pekerjaan)
4     VALUES (NEW.id_karyawan, 1);

```

Kirim

Tutup

## B. Trigger input pekerjaan

Trigger ini adalah sebuah trigger AFTER INSERT dalam SQL. Trigger yang bertujuan untuk memasukkan data baru ke dalam tabel PekerjaanKaryawan setiap kali operasi INSERT terjadi pada tabel yang memicu trigger ini. Nilai kode\_pekerjaan dari baris yang baru saja dimasukkan ke dalam tabel induk diambil menggunakan variabel khusus NEW.kode\_pekerjaan, sementara nilai 1 digunakan sebagai nilai statis untuk id\_karyawan. Dengan demikian, trigger ini memungkinkan untuk mengaitkan pekerjaan baru dengan karyawan yang telah ditentukan sebelumnya secara otomatis setelah ditambahkan ke dalam sistem.

Ubah

Detail

Nama trigger

after\_pekerjaan\_insert

Tabel

pekerjaan

Waktu

AFTER

Kejadian

INSERT

```

1 BEGIN
2   INSERT INTO PekerjaanKaryawan (id_karyawan,
3     kode_pekerjaan)
4     VALUES (1, NEW.kode_pekerjaan);

```

Kirim

Tutup

#### **IV. Kesimpulan**

Pada praktikum ini, mahasiswa telah mempelajari berbagai konsep penting dalam pengelolaan basis data, termasuk pengelolaan record, SQL Subquery, SQL Join, prosedur, fungsi, dan trigger. Pengelolaan record melibatkan penyimpanan, pembaruan, dan penghapusan data secara efisien, sementara SQL Subquery membantu dalam pengambilan data yang lebih kompleks. SQL Join memperkenalkan cara menggabungkan data dari beberapa tabel untuk analisis komprehensif. Prosedur dan fungsi digunakan untuk otomatisasi tugas rutin dan peningkatan efisiensi pengolahan data, sedangkan trigger membantu menjaga integritas dan konsistensi data dengan menjalankan prosedur otomatis sebagai respons terhadap perubahan dalam basis data. Praktikum ini memperkuat pemahaman mahasiswa tentang dasar-dasar pengelolaan basis data dan teknik-teknik SQL penting, mempersiapkan mereka untuk mengatasi tantangan pengelolaan data di dunia nyata.

#### **V. Referensi**

Arsito Ari Kuncoro. 2021. *Pengantar Bahasa Query*: Yayasan DPI

Irmayanti Safitri, 2024, *Mengenal Pengertian Trigger dalam Database*: Nasabamedia

Eri Mardiani, 2020, *Kumpulan Penjelasan SQL* : PT.Elex Media Komputindo

#### **VI. Link database**

[https://drive.google.com/drive/folders/1EcIA7AcroXlm1cd\\_ZBuhOcVvBPOMuTPY](https://drive.google.com/drive/folders/1EcIA7AcroXlm1cd_ZBuhOcVvBPOMuTPY)