

# **MODUL BASIS DATA II**

## **Semester Genap 2023/2024**



**DISUSUN OLEH :**

**SHERLY CHRISTINA, S.KOM., M.KOM**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS PALANGKARAYA**

# **TATA LAKSANA**

## **PRAKTIKUM BASIS DATA II**

### **TATA TERTIB**

1. Praktikan yang lebih dari 1 (satu) kali tidak mengikuti praktikum tidak diperkenankan untuk mengikuti praktikum modul-modul selanjutnya, dan nilai akhir praktikumnya adalah 0 (nol).
2. Praktikan yang berhalangan hadir wajib menghubungi asisten praktikum sebelum sesi dimulai, dan dapat mengikuti praktikum modul yang sama di sesi yang lain; dengan catatan masih terdapat tempat yang kosong di sesi lain tersebut.
3. Batas keterlambatan adalah 15 menit.
4. Sebelum praktikum dimulai praktikan wajib mengumpulkan 2 buah laporan, yaitu: *Laporan Rencana Praktikum*, dan *Laporan Hasil Praktikum* modul sebelumnya. Tanpa mengumpulkan kedua laporan ini praktikan tidak diperkenankan mengikuti praktikum.
5. Segala bentuk kecurangan dan plagiarisme, baik pada laporan maupun test praktikum, akan berakibat pada nilai E sebagai nilai akhir praktikum.

### **SISTEMATIKA LAPORAN**

1. Laporan Rencana Praktikum
  - Sampul depan
  - Tujuan pembelajaran dari modul yang akan dilaksanakan
  - Tugas pendahuluan
  - Hal-hal yang akan dilakukan selama praktikum
2. Laporan Hasil Praktikum
  1. Sampul depan
  2. BAB I Landasan teori yang dipergunakan untuk menyelesaikan tugas praktikum

3. BAB II            Langkah penyelesaian dan pembahasan tugas praktikum
4. BAB III           Kesimpulan
5. BAB IV           Daftar pustaka

## EVALUASI & PENILAIAN

1. Sebelum praktikum dimulai, akan dilaksanakan pre-test untuk modul sesi tersebut.
2. Di akhir semester akan dilaksanakan test akhir praktikum (responsi) yang mencakup materi seluruh modul.
3. Penilaian laporan hasil praktikum:

• Sampul depan	5 %
• BAB I Landasan teori	30 %
• BAB II            Langkah penyelesaian dan pembahasan tugas praktikum	30 %
□ BAB III Kesimpulan	25 %
• BAB IV Daftar pustaka	10 %

---

Total	100 %
-------	-------

4. Penilaian akhir praktikum:

• Pre-test	15 %
• Praktikum	30 %
• Laporan praktikum	30 %
• Responsi	25 %

---

Total	100 %
-------	-------

5. Penilaian akhir mata kuliah:

• Tugas di Kelas + UTS + Praktikum	50 %
------------------------------------	------

• UAS	50 %
-------	------

---

Total	100 %
-------	-------

## Modul I

### Stored Procedure, Function

#### Tujuan

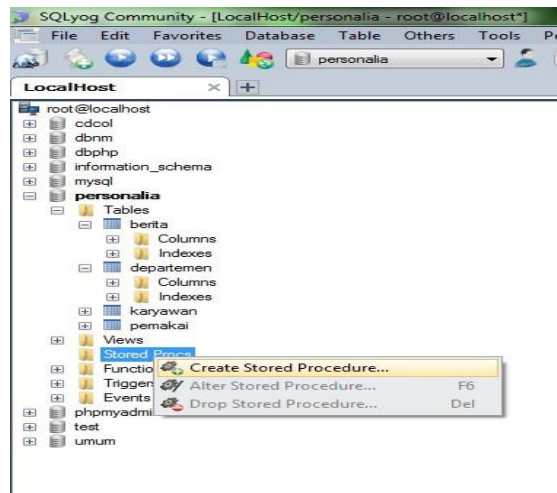
- Mahasiswa memahami manfaat *Stored Procedure* dan mampu membuat *Stored Procedure*.
- Mahasiswa memahami manfaat *Functions* dan mampu membuat *Functions*.

#### Pembahasan

Prosedur pada dasarnya adalah sebuah program yang ditulis dalam bahasa TransactSQL yang disimpan dalam basis data. Prosedur dibentuk dari perintah, variabel, serta alur logik yang terdapat pada SQL. Pada modul ini akan digunakan basis data Personalia yaitu contoh basis data suatu perusahaan yang mengelola data karyawan dan departemen tempat karyawan ditempatkan untuk bekerja. Berikut adalah langkah-langkah pembuatan prosedur pada sebuah tabel Karyawan dari basis data Personalia.

nip	nama	tgl_lahir	jenis_kelamin	kode_dep
12345	Rusli Duchan	1972-12-13	1	1
12346	Sinta Ervianty	1985-03-24	0	2
12347	Rianti Sulaeman	1979-01-13		0 3
12348	Fahri Nasution	1980-02-02		1 4
12349	Karya Utama	1975-07-12		1 5
12350	Burhanudin	1970-11-17		1 1
12351	Jaja Maharaja	1980-02-20		1 2
12352	Dewi Kurniati	1973-08-19		0 3
12353	Saman Hadi	1974-07-30		1 4
12354	Irna Febrianty	1981-06-01		0 2
12355	Lila Mawarni	1980-12-12		0 2
12356	Dika Nurahman	1971-10-03		1 2

1. Pada *folder Stored Procs* klik kanan, pilih **Create Store Procedure**.



2. Masukkan nama *procedure* “Hapus Karyawan” pada kotak dialog **Create Procedure** kemudian klik **Create**.



3. Ketik isi prosedur diantara blok **Begin** dan **End**.

```

1  DELIMITER $$
2
3  CREATE
4      /*[DEFINER = { user | CURRENT_USER }]*/
5      PROCEDURE `personalia`.`HapusKaryawan` (nipkar CHAR(5))
6      /*LANGUAGE SQL
7       | [NOT] DETERMINISTIC
8       | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
9       | SQL SECURITY { DEFINER | INVOKER }
10      | COMMENT 'string'*/
11  BEGIN
12      DELETE FROM karyawan WHERE nip = nipkar;
13  END$$
14
15  DELIMITER ;
16

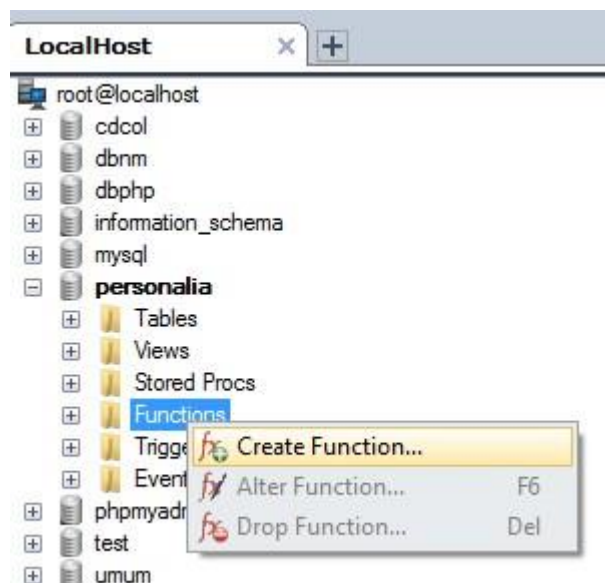
```

4. Kemudian klik .

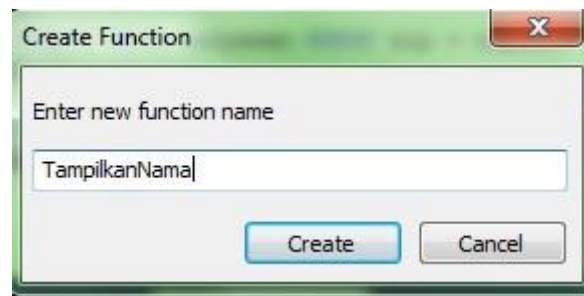
5. Berikut adalah query untuk memanggil prosedur HapusKaryawan CALL  
HapusKaryawan('123456');

Seperti halnya *Store Procedure*, *Function* juga merupakan program yang ditulis dalam bahasa Transact-SQL dan disimpan dalam basis data, tetapi *Function* dapat memberikan nilai balik (*Return Value*). Berikut adalah langkah-langkah untuk membuat *Function* pada tabel Karyawan dari basis data Personalia.

1. Pada *folder Functions* klik kanan, pilih **Create Function**.



6. Masukkan nama *fungsi* “TampilkanNama” pada kotak dialog **Create Function** kemudian klik **Create**.



```
Query History HapusKaryawan TampilkanNama x +
1 DELIMITER $$
2
3 CREATE
4 /*[DEFINER = { user | CURRENT_USER }]*/
5 FUNCTION `personalia`.`TampilkanNama`()
6 RETURNS TYPE
7 /*LANGUAGE SQL
8 | [NOT] DETERMINISTIC
9 | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
10 | SQL SECURITY { DEFINER | INVOKER }
11 | COMMENT 'string'*/
12 BEGIN
13
14 END$$
15
16 DELIMITER ;
```

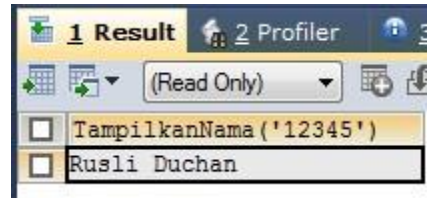
2. Isi fungsi diantara blok **Begin** dan **End**.

```
Query History HapusKaryawan TampilkanNama x +
1 DELIMITER $$
2
3 CREATE
4 /*[DEFINER = { user | CURRENT_USER }]*/
5 FUNCTION `personalia`.`TampilkanNama`(nipkar CHAR (5))
6 RETURNS CHAR (35)
7 /*LANGUAGE SQL
8 | [NOT] DETERMINISTIC
9 | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
10 | SQL SECURITY { DEFINER | INVOKER }
11 | COMMENT 'string'*/
12 BEGIN
13     DECLARE namakar CHAR(35); -- mendeklarasikan variabel lokal
14     SELECT nama FROM karyawan WHERE nip=nipkar INTO namakar;
15     RETURN namakar;
16 END$$
17
18 DELIMITER ;
```



3. Kemudian Klik 

4. Berikut adalah query untuk memanggil Function TampilkanNama. SELECT TampilkanNama('12345');



## Tugas Praktikum modul 1

1. Buat database sesuai dengan ERD yang telah disediakan (Tentukan tipe data yang ideal bagi tiap field dan perhatikan relasi antar tabel)

Isi data pada tiap-tiap tabel minimal 5 record KECUALI jabatan dimana ada 3 record (MANAJER, KASIR, MONTIR).

1. Buat 2 Stored Procedure untuk Insert data Pegawai dan Nomor\_pegawai secara bersamaan dan Transaksi dan Header\_transaksi secara bersamaan.
2. Buat Stored Procedure untuk Insert data pada tabel lainnya.
3. Buat Stored Procedure untuk Delete pada semua tabel.
4. Buat Stored Procedure untuk Update data Pegawai dan Nomor\_pegawai secara bersamaan.
5. Buat Stored Procedure untuk Update data pada tabel lainnya.
6. Buat Fungsi untuk menampilkan semua transaksi seorang pegawai pada tanggal tertentu
7. Buat Fungsi untuk menampilkan semua Sparepart dan semua Service yang dibeli pada suatu transaksi

**Catatan:** Silakan referensi ERD dibawah, dan simpanlah tugas-tugas yang telah dikerjakan dengan baik, karena tugas setiap modul pada praktikum ini saling terkait.

## MODUL II

### MENGGUNAKAN FUNGSI-FUNGSI AGREGASI, FUNGSI- FUNGSI STRING DAN MANAJEMEN TRANSAKSI

#### Tujuan

- Mahasiswa dapat menggunakan fungsi-fungsi agregasi dan fungsi-fungsi string dalam Stored Procedure dan Function
- Mahasiswa memahami manfaat Manajemen Transaksi.
- Mahasiswa dapat membuat Manajemen Transaksi.

#### Pembahasan

##### Aritmatik

Penghitungan aritmatika dapat dilakukan dengan menggunakan \*/+- seperti umumnya. Selain itu penghitungan aritmatika juga dilengkapi oleh fungsi-fungsi agregasi. Berikut adalah beberapa fungsi agregasi.

- **AVG**  
Fungsi untuk mencari rata-rata.
- **MAX**  
Fungsi untuk mencari maksimum.
- **MIN**  
Fungsi untuk mencari minimum

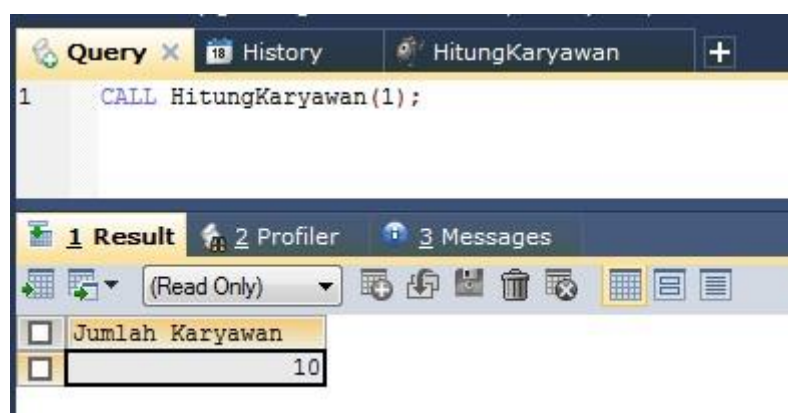
##### □ **COUNT**

Fungsi untuk menghitung record dengan kualifikasi tertentu.

Contoh penggunaan fungsi agregasi seperti berikut ini. Pada prosedur **HitungKaryawan** digunakan fungsi Count untuk menghitung jumlah karyawan yang berkerja di suatu departemen.

```
Query History HitungKaryawan +
1 DELIMITER $$
2
3 USE `personalia`$$
4
5 DROP PROCEDURE IF EXISTS `HitungKaryawan`$$
6
7 CREATE DEFINER=`root`@`localhost` PROCEDURE `HitungKaryawan`(kodeDep CHAR(1))
8 BEGIN
9     SELECT COUNT(NIP) AS 'Jumlah Karyawan' FROM Karyawan WHERE kode_Dep = kodeDep ;
10 END$$
11
12 DELIMITER ;
```

Prosedur **HitungKaryawan** akan menghitung jumlah karyawan yang berkerja pada departemen yang kode departemennya adalah 1.



## String pada SQL

Berikut adalah fungsi-fungsi yang digunakan untuk mengolah data bertipe String.

- Substring

Fungsi Substring mengembalikan sebagian baik karakter atau string biner, atau string.

Fungsi Substring terdiri atas tiga parameter:

1. Sebuah karakter atau string biner, nama kolom, atau string-ekspresi bernilai yang mencakup nama kolom.
2. Posisi di mana substring harus dimulai.
3. Panjang (dalam jumlah karakter, atau dalam jumlah byte untuk biner) dari string yang akan dikembalikan.

- Replace

Fungsi Replace digunakan untuk mengganti teks (satu atau kumpulan karakter) tertentu.

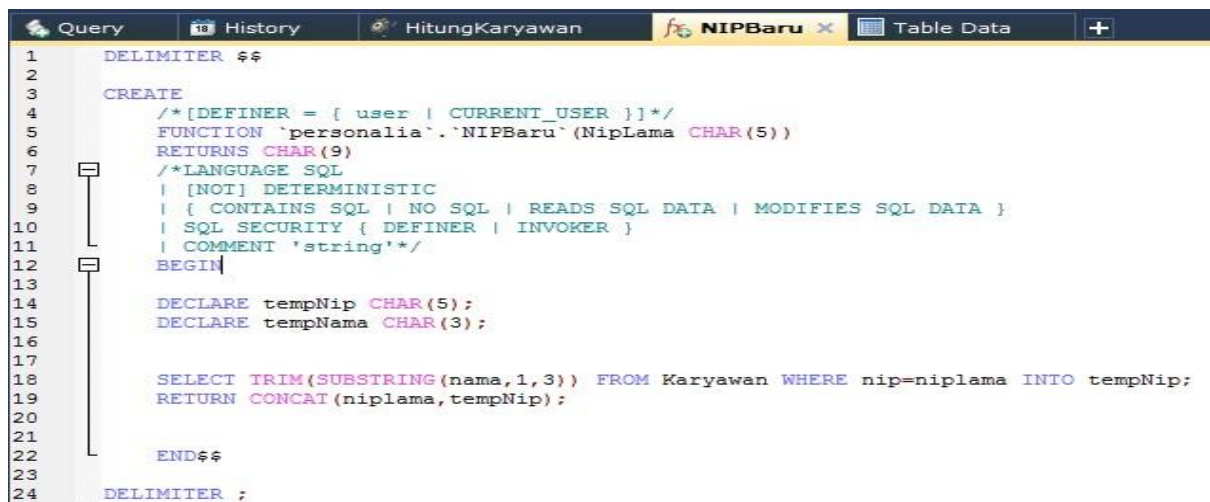
- Concat

Fungsi Concat digunakan untuk menggabungkan satu atau lebih teks yang diberikan.

- Trim

Fungsi Trim untuk menghilangkan atau menghapus spasi dari bagian kiri dan kanan suatu string.

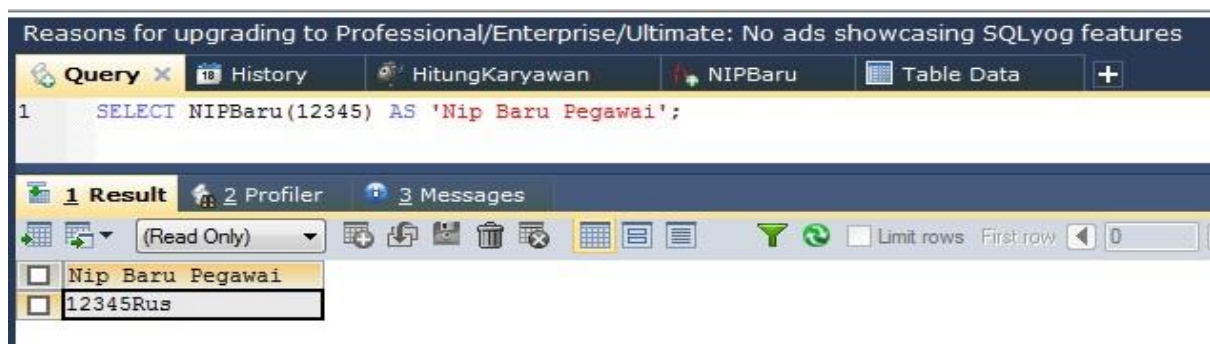
Contoh penggunaan fungsi string seperti berikut ini. Pada fungsi **NIPBaru** akan digunakan beberapa fungsi string untuk membuat nip baru karyawan, yang terdiri atas gabungan dari Nip lama dengan tiga karakter pertama dari nama karyawan.



```

1  DELIMITER $$
2
3  CREATE
4  /*[DEFINER = { user | CURRENT_USER }]*/
5  FUNCTION `personalia`.`NIPBaru` (NipLama CHAR(5))
6  RETURNS CHAR(9)
7  /*LANGUAGE SQL
8  | [NOT] DETERMINISTIC
9  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
10 | SQL SECURITY { DEFINER | INVOKER }
11 | COMMENT 'string'*/
12 BEGIN
13
14 DECLARE tempNip CHAR(5);
15 DECLARE tempNama CHAR(3);
16
17
18 SELECT TRIM(SUBSTRING(nama,1,3)) FROM Karyawan WHERE nip=niplama INTO tempNip;
19 RETURN CONCAT (niplama,tempNip);
20
21
22 END$$
23
24 DELIMITER ;
  
```

Fungsi **NIPBaru** akan membuat Nip Baru bagi karyawan yang nip lamanya adalah 12345, dengan cara menggabungkan nip lama karyawan dengan 3 karakter pertama dari nama karyawan tersebut



```

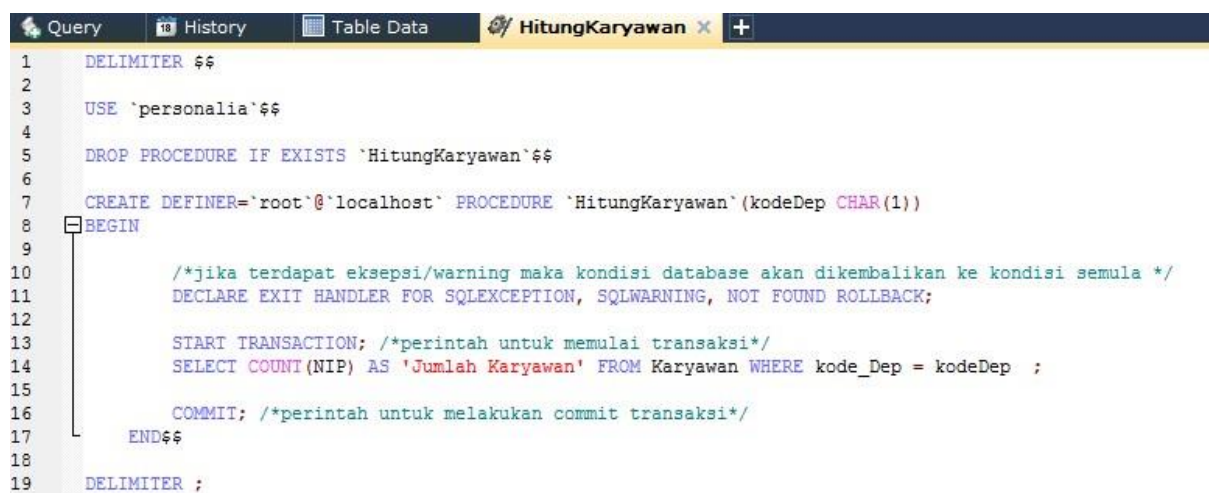
1  SELECT NIPBaru(12345) AS 'Nip Baru Pegawai';
  
```

1 Result	
Nip Baru Pegawai	12345Rus

## Manajemen Transaksi

Sistem basis data harus menjamin ACID (*Atomicity, Consistency, Isolation and Durability*) pada setiap transaksi. Sebuah transaksi mungkin membutuhkan beberapa *query*, yang membaca atau menulis informasi dalam basis data. Ketika hal ini terjadi, harus dipastikan tidak ada *query* yang terlewat yang akan digunakan pada basis data. Terdapat dua pilihan transaksi yang bisa dilakukan, yaitu *commit transaction* atau *rollback transaction*. Namun sebelumnya harus didahului dengan perintah *begin transaction* atau *start transaction*.

Berikut adalah contoh implementasi manajemen transaksi pada prosedur **HitungKaryawan**.



```
1  DELIMITER $$
2
3  USE `personalia`$$
4
5  DROP PROCEDURE IF EXISTS `HitungKaryawan`$$
6
7  CREATE DEFINER=`root`@`localhost` PROCEDURE `HitungKaryawan`(kodeDep CHAR(1))
8  BEGIN
9
10     /*jika terdapat eksepsi/warning maka kondisi database akan dikembalikan ke kondisi semula */
11     DECLARE EXIT HANDLER FOR SQLEXCEPTION, SQLWARNING, NOT FOUND ROLLBACK;
12
13     START TRANSACTION; /*perintah untuk memulai transaksi*/
14     SELECT COUNT(NIP) AS 'Jumlah Karyawan' FROM Karyawan WHERE kode_Dep = kodeDep ;
15
16     COMMIT; /*perintah untuk melakukan commit transaksi*/
17 END$$
18
19 DELIMITER ;
```

## Tugas Praktikum modul 2

Kerjakan tugas-tugas berikut, dengan menggunakan database bengkel mobil seperti pada Modul I.

1. Buat Procedure untuk menampilkan Sparepart dan Service yang paling banyak dibeli.
2. Buat Procedure untuk menampilkan harga Sparepart dan Service yang paling mahal.
3. Buat Procedure untuk menampilkan harga Sparepart dan Service paling murah.
4. Buat Procedure untuk menghitung banyaknya transaksi pembelian sparepart.
5. Buat Fungsi untuk membuat id\_pembelian\_sparepart dengan menggabungkan 3 huruf pertama nama **Pelanggan** dengan **id\_pelanggan**, dan gabungkan 3 huruf pertama nama

**Pegawai** dengan **tanggal** dilakukannya transaksi dan juga tambahkan 5 angka random di akhir. Misalnya Pelanggan (Dodi, 123) = (dod123), (Andre, 12-03-2024) = (and12032024), angka random (35249) menjadi (dod123and1203202435249)

6. Buat Fungsi untuk membuat id\_pembelian\_service dengan menggabungkan 3 huruf pertama nama **Pelanggan** dengan **id\_transaksi**, dan gabungkan 3 huruf pertama nama **Pegawai** dengan **id\_pelanggan** dan juga tambahkan 5 angka random di akhir. Misalnya Pelanggan (Dodi, 123) = (dod123), (Andre, 12-03-2024) = (and12032024), angka random (35249) menjadi (dod123and1203202435249)

## MODUL III

### VIEW

#### Tujuan

- Mahasiswa dapat mengetahui manfaat *view*.
- Mahasiswa dapat membuat *view* pada mysql

#### Pembahasan

*View* adalah objek di dalam database yang berisi kumpulan kolom yang dihasilkan dari Perintah *select*. Dengan kata lain yang lebih sederhana, *view* adalah *object* yang menyimpan hasil *query*, baik dari satu tabel atau lebih, didalam *database* *view* juga sering dinamakan sebagai “tabel virtual” , karena *view* sebenarnya tidak memiliki data. Data yang ditampilkan oleh sebuah *view* diambil dari tabel-tabel aktual yang disertakan dalam *SELECT*.



The screenshot shows a MySQL database interface. At the top, there is an SQL query: `SELECT * FROM baju;`. Below the query, the result is displayed in a table with three columns: `nobaju`, `ukuran_baju`, and `kapasitas_pesan`. The table contains four rows of data.

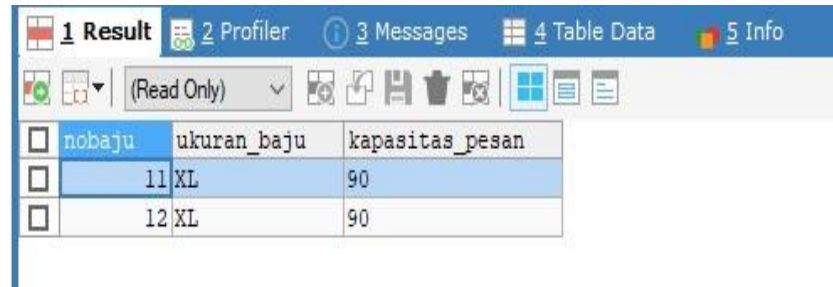
nobaju	ukuran_baju	kapasitas_pesan
11	XL	90
12	XL	90
13	L	80
14	M	80

Gambar diatas adalah contoh tabel ukuran baju yang akan digunakan untuk membuat *view*. Untuk membuat *view* langkah langkah yang perlu dilakukan :

1. Tuliskan perintah Create View **nama tabel baru** AS Select **kolom 1,kolom 2,...** From **nama tabel yang ada** Where **kondisi**;  
**Contoh :** CREATE VIEW ukuranbaju AS SELECT nobaju,ukuran\_baju,kapasitas\_pesan FROM baju Where ukuran\_baju = 'XL' ;  

```
CREATE VIEW ukuranbaju AS SELECT nobaju,ukuran_baju,kapasitas_pesan FROM baju WHERE ukuran_baju = 'XL' ;
```
2. Untuk melihat apakah *view* dapat digunakan maka kita gunakan perintah Select \* from **namakolomview**. Contoh : Select \* From ukuranbaju;

```
SELECT * FROM ukuranbaju;
```



The screenshot shows the 'Result' tab in SQL Server Enterprise Manager. The table has three columns: 'nobaju', 'ukuran\_baju', and 'kapasitas\_pesan'. The data is as follows:

nobaju	ukuran_baju	kapasitas_pesan
11	XL	90
12	XL	90

3. Untuk menghapus view gunakan syntax *Drop view [view\_name];*

### Tugas Praktikum modul 3

1. Buatlah View bernama **ban\_dunlop** untuk menampilkan data-data sparepart beserta transaksi dari sparepart yaitu ban bernama Dunlop
2. Buatlah View bernama **pemasukan\_pegawai\_service** menampilkan data dari setiap pegawai yang berjabatan montir beserta pemasukan dari total hasil pekerjaan service dia.
3. Buatlah View bernama **daftar\_pegawai\_** untuk menampilkan data pegawai beserta jabatan dan nomor pegawai.
4. Buatlah View bernama **detail\_transaksi** untuk menampilkan nama pelanggan, nama pegawai, jumlah sparepart yang dibeli, jumlah service yang di beli, dan tanggal pembelian.



## MODUL IV

### TRIGGER

#### Tujuan

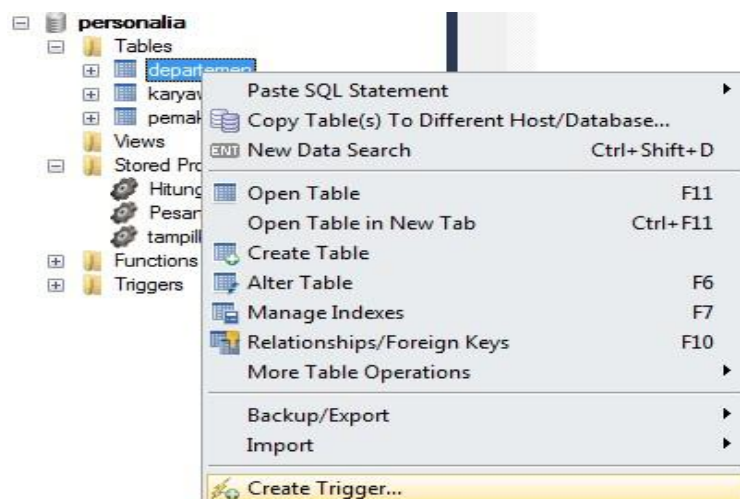
- Mahasiswa mampu memahami trigger serta penggunaannya untuk optimasi sebuah *database*.

#### Pembahasan

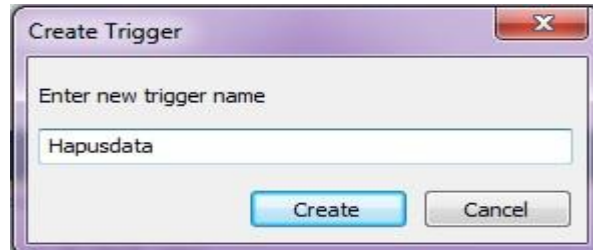
Trigger adalah sekumpulan perintah SQL yang secara otomatis dijalankan apabila ada perintah INSERT, UPDATE, atau DELETE yang dijalankan di dalam tabel. Aplikasi yang dapat dilakukan oleh trigger di antaranya adalah:

- Membuat isi dari kolom yang diambil dari kolom yang lain.
- Membuat mekanisme validasi yang mencakup query pada banyak tabel.
- Membuat log untuk mendaftarkan penggunaan tabel.
- Meng-*update* tabel-tabel lain apabila ada penambahan atau perubahan lain di dalam tabel yang sedang aktif.

Langkah-langkah untuk membuat trigger ditunjukkan pada gambar di bawah ini. Klik kanan pada nama table, pilih **Create Trigger**.



Kemudian isi nama Trigger pada kotak dialog **Create Trigger**.



Berikut adalah contoh trigger bernama **HapusData**. Pengaktifan trigger akan tergantung pada *event* yang dipilih (BEFORE/AFTER INSERT/UPDATE/DELETE). Trigger ini akan dijalankan ketika sebuah *record* pada tabel departemen dihapus. Trigger ini akan secara otomatis menghapus semua data karyawan yang bekerja pada departemen yang telah terhapus datanya.

```
Query | History | PesanHapus | Table Data | HapusData x +
1  DELIMITER $$
2
3  CREATE
4      /*[DEFINER = { user | CURRENT_USER }]*/
5      TRIGGER `personalia`.`HapusData` AFTER DELETE
6      ON `personalia`.`departemen`
7      FOR EACH ROW BEGIN
8          DELETE FROM karyawan WHERE kode_dep = old.kode_dep;
9      END$$
10
11  DELIMITER ;
```

#### Tugas Praktikum modul 4

1. Buat trigger bernama **UpdateTotalBiayaAIPembelianService** dimana setelah operasi insert pada pembelian service maka total biaya akan ditambahkan pada tabel transaksi.
2. Buat trigger bernama **UpdateTotalBiayaAUPembelianService** dimana setelah operasi update pada pembelian service maka total biaya akan diperbaharui sesuai dengan perubahan di pembelian service pada tabel transaksi.
3. Buat trigger bernama **UpdateTotalBiayaADPembelianService** dimana setelah operasi delete pada pembelian service maka total biaya akan dikurang pada tabel transaksi

4. Buat trigger bernama **UpdateTotalBiayaAIPembelianSparepart** dimana setelah operasi insert pada pembelian sparepart maka total biaya akan ditambahkan pada tabel transaksi.
5. Buat trigger bernama **UpdateTotalBiayaAUPembelianSparepart** dimana setelah operasi update pada pembelian sparepart maka total biaya akan diperbaharui sesuai dengan perubahan di pembelian sparepart pada tabel transaksi.
6. Buat trigger bernama **UpdateTotalBiayaADPembelianSparepart** dimana setelah operasi delete pada pembelian sparepart maka total biaya akan dikurang pada tabel transaksi.
7. Buat trigger bernama **UpdateJumlahStokAIPembelianSparepart** dimana setelah insert pada pembelian sparepart maka stok akan diperbaharui
8. Buat trigger bernama **UpdateJumlahStokAUPembelianSparepart** dimana setelah update pada pembelian sparepart maka stok akan diperbaharui.
9. Buat trigger bernama **UpdateJumlahStokADPembelianSparepart** dimana setelah delete pada pembelian sparepart maka stok akan diperbaharui.

## Modul V

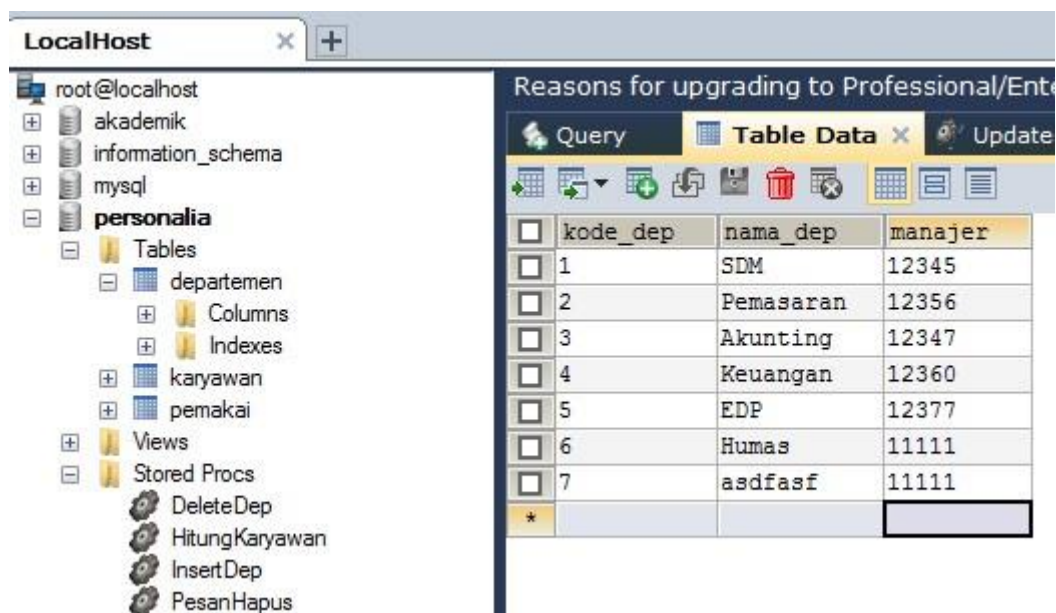
### Penggunaan Function dan Stored Procedure pada Web Sederhana

#### Tujuan

- Mahasiswa dapat menggunakan *Function* dan *Stored Procedure* pada aplikasi Web sederhana.

#### Pembahasan

Pada Modul ke-5 diberikan contoh penggunaan *procedure* pada sebuah aplikasi web sederhana. Tabel yang digunakan pada contoh ini adalah tabel Departemen dari database Personalia.



The screenshot shows the MySQL Workbench interface. On the left, the 'personalia' database is selected, showing its structure: Tables (departemen, karyawan, pemakai), Views, and Stored Procs (DeleteDep, HitungKaryawan, InsertDep, PesanHapus). On the right, the 'Table Data' tab is active, displaying the data for the 'departemen' table.

	kode_dep	nama_dep	manajer
<input type="checkbox"/>	1	SDM	12345
<input type="checkbox"/>	2	Pemasaran	12356
<input type="checkbox"/>	3	Akunting	12347
<input type="checkbox"/>	4	Keuangan	12360
<input type="checkbox"/>	5	EDP	12377
<input type="checkbox"/>	6	Humas	11111
<input type="checkbox"/>	7	asdfasf	11111
<input type="checkbox"/>	*		

Prosedur yang digunakan adalah prosedur **DeleteDep** dan **InsertDep**. Prosedur DeleteDep adalah prosedur untuk menghapus data dari tabel departemen.

```
Query Table Data DeleteDep x +
1 DELIMITER $$
2
3 USE `personalia`$$
4
5 DROP PROCEDURE IF EXISTS `DeleteDep`$$
6
7 CREATE DEFINER=`root`@`localhost` PROCEDURE `DeleteDep` (kodedep CHAR(1))
8 BEGIN
9     DELETE FROM departemen WHERE kode_dep=kodedep;
10 END$$
11
12 DELIMITER ;
```

Sedangkan prosedur InsertDep digunakan untuk menambah data ke dalam tabel Departemen.

```
Query Table Data DeleteDep InsertDep x +
1 DELIMITER $$
2
3 USE `personalia`$$
4
5 DROP PROCEDURE IF EXISTS `InsertDep`$$
6
7 CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertDep` (kodedep CHAR(1), namadep CHAR(15),
8 manajerdep CHAR(5))
9 BEGIN
10     INSERT INTO departemen VALUES (kodedep,namadep,manajerdep);
11 END$$
12
13 DELIMITER ;
```

Kemudian berikut ini adalah kode program dalam bahasa php untuk melakukan tambah dan hapus data pada tabel departemen.

## fungsi.php

```
<?php
```

```
global $koneksi;
```

```
function connect()
```

```
{
```

```
    $host="localhost";
```

```
    $username = "root";
```

```
    $password="";
```

```
    $database="personalia";
```

```
    $koneksi = mysqli_connect($host,$username,$password);
```

```
    if(!$koneksi)
```

```
{
```

```

        die('koneksi gagal:'.mysql_error());
    }

}

function query($sql)
{
    $result=mysqli_query($koneksi,$sql);
    return $result;
}

function deleteDep($kodedep)
{
    $result = mysqli_query($koneksi,"call DeleteDep('$kodedep')");
    return $result;
}

function insertDep($kodedep,$namadep,$manajerdep)
{
    $result = mysqli_query($koneksi,"call InsertDep('$kodedep','$namadep','$manajerdep')");
    return $result;
}

function updateDep($kodedep,$namadep,$manajerdep)
{
    $result =mysqli_query($koneksi,"call UpdateDep('$kodedep','$namadep','$manajerdep')"); return
    $result;
}
?>

```

## departemen.php

```

<?php include("../fungsi.php");
$action="";
$kode_dep="";
$nama_dep="";
$manajer="";

```

```

if (isset($_GET['action']))
{
    $action = $_GET['action'];
    $kode_dep= $_GET['kode_dep'];
    $nama_dep= $_GET['nama_dep'];
    $manajer= $_GET['manajer'];
}
connect();
$result = query("Select * From departemen");
?>

```

```

<html>
<head>
<title>Departemen</title>
</head>
<body>
<form method='post' action='insert.php'>
<table>
<tr>
<td>Kode Departemen</td>
<td><input type='text' name='kode_dep' value='<?php print $kode ?>' /> </input>
</td>
</tr>
<tr>
<td>Nama Departemen </td>
<td><input type='text' name='nama_dep' value='<?php print $nama ?>' /> </input>
</td>
</tr>
<tr>
<td>Manajer Departemen</td>
<td><input type='text' name='manajer' value='<?php print $manajer ?>' /> </input>
</td>
</tr>
<tr>
<td></td>
<td align='right'><input type='submit' value='Save' /> </input>
<input type='hidden' value='<?php print $action ?>' name='action' /> </input>
</td>

```

```

        </tr>
    </table>
</form>
<br>
<br>
<table border=1>
    <tr>
        <th>Kode Departemen </th>
        <th>Nama Departemen </th>
        <th>Manajer Departemen </th>
        <th>Delete</th>
    </tr>
<?php
$action="";

While($row=mysql_fetch_object($result)){
print  ("<tr>
        <td>$row->kode_dep</td>
        <td>$row->nama_dep</td>
        <td>$row->manajer</td>
        <td><a href = 'delete.php?kode_dep=$row->kode_dep'>Delete</a></td>
    </tr>");
}
?>
</table>
</body>
</html>

```

## insert.php

```

<?php

if($_POST['action']=="")
{
    $kode_dep = $_POST['kode_dep'];
    $nama_dep = $_POST['nama_dep'];
    $manajer = $_POST['manajer'];

```



```

include("../fungsi.php"); connect();
insertDep($kode_dep,$nama_dep,$manajer);
header('Location:departemen.php');
}
?>

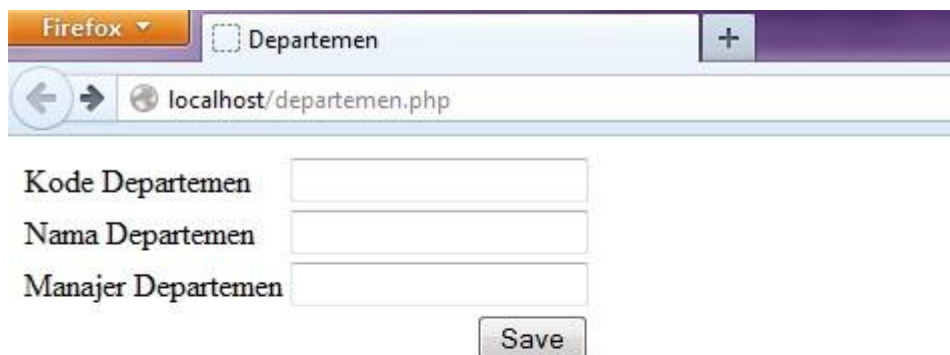
```

## delete.php

```

<?php
if(isset($_GET['kode_dep']))
{
    $kode_dep = $_GET['kode_dep'];
include("../fungsi.php"); connect();
deleteDep($kode_dep);
header('Location:departemen.php');
}
?>

```



Firefox ▾ Departemen +

localhost/departemen.php

Kode Departemen

Nama Departemen

Manajer Departemen

Save

Kode Departemen	Nama Departemen	Manajer Departemen	Delete
1	SDM	12345	<a href="#">Delete</a>
2	Pemasaran	12356	<a href="#">Delete</a>
3	Akunting	12347	<a href="#">Delete</a>
4	Keuangan	12360	<a href="#">Delete</a>
5	EDP	12377	<a href="#">Delete</a>
6	Humas	11111	<a href="#">Delete</a>

## **Tugas**

Buat aplikasi web sederhana tanpa style untuk operasi CRUD pada database yang telah anda buat. Gunakan Stored Procedure, Function dan View yang telah anda buat..

## ERD Bengkel Mobil

