



**MILIK ITI**

**UNTUK KALANGAN SENDIRI**

**TIDAK DIPERJUAL BELIKAN**

# **BUKU PETUNJUK PRAKTIKUM BASIS DATA I**

Disusun oleh :

**Tim Asisten**

**LABORATORIUM  
PRODI INFORMATIKA  
INSTITUT TEKNOLOGI INDONESIA**

## **LABORATORIUM PRODI INFORMATIKA - ITI**

### **TATA TERTIB PRAKTIKUM KOMPUTER**

#### **A. Penilaian Praktikum ini dilakukan terhadap 3 hal yaitu :**

1. Penilaian Hasil Praktikum
2. Kehadiran
3. Sikap dan tingkah laku Praktikan.

#### **B. Dalam Praktikum ini Praktikan harus memperhatikan hal-hal sebagai berikut :**

1. Kehadiran Praktikan harus 100% semua modul praktikum harus dilaksanakan. Apabila Praktikan tidak hadir pada jadwal yang telah ditentukan, praktikum harus dilaksanakan pada jam yang lain dengan konsekuensi membayar biaya perawatan sebesar Rp. 20.000,- (dua puluh ribu rupiah) permodul. Praktikan yang tidak melaksanakan salah satu modul dan tidak mengganti pada jam yang lain, maka nilai praktikum modul tersebut adalah 0 (nol) dan batas waktu menyisip seminggu, sebelum praktikum berikutnya berjalan.
2. Bagi yang memerlukan praktikum tambahan, dapat menggunakan fasilitas laboratorium (bila ada yang kosong) dengan membayar biaya perawatan sebesar Rp. 5000,- (lima ribu rupiah) perjam atau bagian dari 1 jam.
3. Keterlambatan lebih dari 15 menit tidak diperkenankan mengikuti praktikum.
4. Setiap praktikum, Kartu Praktikum harus dibawa dan diserahkan pada asisten yang bertugas.
5. Duduklah pada tempat yang telah ditentukan sesuai dengan nomor yang tertera di Kartu Praktikum.
6. Tas, Buku, Disket, Flash Disk/USB, dll harus diletakkan pada tempat/loker yang telah disediakan kecuali Buku Petunjuk Praktikum.
7. Praktikan harus berpakaian rapi dan tidak diperkenankan memakai sandal, mengenakan headset.
8. Bersihkan sepatu dari debu / tanah sebelum memasuki ruang praktikum.
9. Ruangan Praktikum merupakan Ruangan ber-AC. Dimana tidak seorangpun diperkenankan merokok, membawa makanan dan minuman.
10. Peralatan Komputer yang ada adalah peralatan yang mahal. Kecerobohan Praktikan yang mengakibatkan kerusakan alat harus ditanggung penggantianannya oleh Praktikan sendiri.
11. Praktikan dilarang mengganti/merubah-ubah Konfigurasi, Software atau Password yang sudah ada.
12. Selama praktikum berlangsung jagalah sopan santun dan ketenangan agar praktikum yang saudara ikuti dapat bermanfaat. Sangsi atas salah satu pelanggaran ini akan mempengaruhi nilai praktikum saudara.
13. Setelah praktikum selesai lingkungan kerja harus bersih dan rapi seperti semula. Setelah selesai kerja setiap praktikan harus membuang kertas yang tidak dipakai dan sampah lainnya ketempat yang disediakan dan merapihkan kursi kembali.

14. Sebelum meninggalkan ruangan, Komputer dan Monitor, Saudara harus dalam keadaan OFF (mati).
15. Harus meninggalkan ruangan apabila bel tanda waktu praktikum selesai ber-bunyi.
16. Nilai Praktikum merupakan komponen penentu nilai akhir.

### **TATA TERTIB PENGGUNAAN PETUNJUK PRAKTIKUM**

Buku Petunjuk Praktikum adalah milik Institut, tidak diberikan tetapi dipinjamkan saat praktikum selama satu semester. Apabila rusak/hilang maka Praktikan dikenakan denda sebesar Rp. 50.000,- (Lima puluh ribu rupiah) atau mengganti buku tersebut.

### **RIWAYAT PERUBAHAN**

Tanggal	Versi	Penulis	Keterangan
01 Maret 2021	Versi 1	Sumiarti	Pembuatan awal
25 Maret 2021	Versi 2	Figri Maulana	Penulisan Ulang
30 Maret 2021	Versi 2.5	Muhamad Ramli	Koreksi dan penambahan modul

# DAFTAR ISI

<b>MODUL 1</b>	1
<b>PENGENALAN MYSQL/MARIADB</b>	1
1. Memulai MySQL / MariaDB	1
2. Membuat Database	1
3. Mengaktifkan Database	2
4. Membuat Tabel	2
5. Tugas 1.1	3
6. Mengisi Data Dalam Tabel	4
7. Menampilkan Data dari Tabel	5
8. Pengubahan Struktur Tabel	5
9. Melihat Struktur Tabel	6
10. ERD (Entity Relationship Diagram)	7
11. Menghapus Tabel	7
12. Tugas 1.2	7
<b>MODUL 2</b>	9
<b>Tipe Data, Operator Aritmatika</b>	9
1. Operator Aritmatika	9
2. Penggunaan Operator Aritmatika Tanpa Tabel	9
3. Penggunaan Operator Aritmatika Dalam Tabel	10
<b>MODUL 3</b>	14
<b>Membuat dan Memanipulasi Tabel</b>	14
1. Membuat Tabel	14
2. Constraint	15
3. Manipulasi Tabel	16
3.1. Menambahkan kolom baru → alter + add	16
3.2. Menghapus kolom → alter + drop	16
3.3. Memodifikasi Kolom → alter + change	16
3.4. Melihat Daftar Database dan Tabel	17
<b>MODUL 4</b>	19
<b>Memanipulasi Data</b>	19
1. Menambah Data	19
2. Tugas 4.1	20
3. Mengubah Data	21
4. Menghapus Data	22
4.1. Menghapus sebuah baris	22
5. Tugas 4.2	23
<b>MODUL 5</b>	24
<b>Query Dasar</b>	24
1. Query	24
2. Tugas 5.1	24
3. Menampilkan Data Tanpa Syarat	25
4. Tugas 5.2	26
5. Menampilkan Data Bersyarat	26
6. Tugas 5.3	27
7. Query Beberapa Tabel	27
8. Tugas 5.4	27
<b>MODUL 6</b>	28
<b>Query Lanjutan dan Fungsi Agregat</b>	28
1. Operator BETWEEN	28
2. Operator IN	28
3. Operator LIKE	28
4. Mengurutkan Data (ORDER BY)	29
5. Klausa Limit	29

6.	Fungsi Agregat.....	30
7.	Tugas 6.1.....	30
8.	Mengelompokkan Data (GROUP BY) .....	31
9.	Tugas 6.2.....	31
<b>MODUL 7</b>	.....	<b>32</b>
<b>Fungsi</b>	.....	<b>32</b>
1.	Pendahuluan.....	32
2.	Fungsi String.....	32
3.	Tugas 7.1.....	34
4.	Fungsi tanggal dan Waktu.....	34
5.	Fungsi Numerik .....	36
6.	Tugas 7.2.....	37
7.	Fungsi Yang Lain.....	37
<b>MODUL 8</b>	.....	<b>38</b>
<b>JOIN</b>	.....	<b>38</b>
1.	Pendahuluan.....	38
2.	Membuat Database, Tabel dan Mengisi data .....	38
3.	Inner Join .....	39
4.	Left Join .....	40
5.	Right Join.....	41
<b>MODUL 9</b>	.....	<b>42</b>
<b>STORE PROCEDURE</b>	.....	<b>42</b>
1.	Pendahuluan.....	42
2.	Membuat Database dan Tabel Sederhana .....	42
3.	Membuat Store Procedure.....	43
4.	Store Procedure dengan Parameter .....	43
5.	Data Manipulasi dengan Store Procedure .....	44
6.	Tugas.....	45
<b>MODUL 10</b>	.....	<b>46</b>
<b>VIEW</b>	.....	<b>46</b>
1.	Pendahuluan.....	46
2.	Membuat VIEW.....	46
3.	Menghapus VIEW .....	47
4.	Tugas.....	47

## MODUL 1

### PENGENALAN MYSQL/MARIADB

---

#### A. Tujuan Belajar

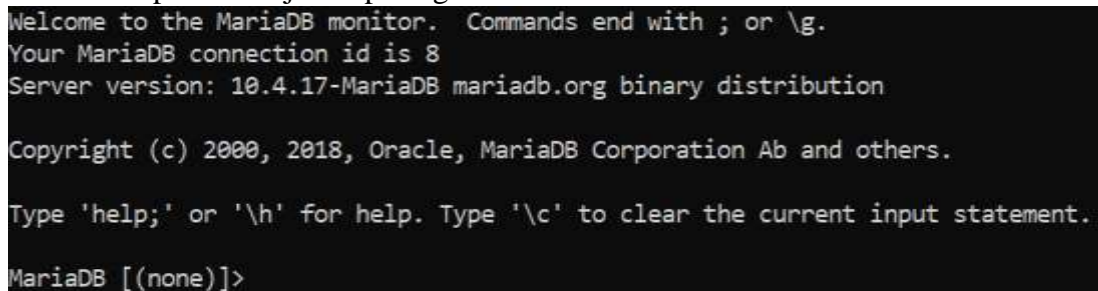
Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami cara membuat database dan sintaks dasar pada MySQL dan MariaDB
2. Memahami tipe data yang ada pada MySQL dan MariaDB
3. Memahami Entity Relationship Diagram (ERD)

#### B. Praktikum

##### 1. Memulai MySQL / MariaDB

Dalam praktikum ini akan digunakan MySQL atau MariaDB sebagai utilitas untuk membangun database. Untuk memulai database maka layanan database harus berjalan terlebih dahulu, jalankan misalnya pakai xampp atau service mandiri database yang terinstall dikomputer. kemudian pada konsol / command prompt CMD dan ketik pada konsol > **mysql**. Layar tampilan yang didapatkan jika Anda masuk ke dalam konsol database seperti ditunjukkan pada gambar 1.1:



```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.17-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

*Gambar 1.1 Tampilan Konsol Database*

Selain melalui konsol untuk memasuki database, Anda juga bisa menggunakan phpMyAdmin berbasis web yang bisa diakses melalui browser, jika Anda menjalankan service XAMPP/WAMP. Selain itu Anda bisa menggunakan akses dengan GUI menggunakan aplikasi seperti SQLYog, HeidiSQL, DBeaver, MySQL Workbench dan lainnya. Dalam praktikum ini akan lebih banyak menggunakan perintah SQL pada konsol atau pada query editor jika menggunakan tools memiliki GUI.

##### 2. Membuat Database

Lanjutkan dengan pembuatan database akademik yang berisi data-data tentang mahasiswa, dosen, matakuliah, krs dan jadwal.

Untuk membuat database akademik digunakan perintah :

```
CREATE DATABASE akademik;
```

Dilayar akan ditampilkan

```
Query OK, 1 row affected (0.008 sec)
```

Tampilan ini menunjukkan bahwa database akademik telah terbentuk.

### 3. Mengaktifkan Database

Sebelum melakukan operasi pada database akademik maka terlebih dahulu harus database harus dipilih, dengan menggunakan perintah :

**USE akademik;**

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.17-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database akademik;
Query OK, 1 row affected (0.008 sec)

MariaDB [(none)]> use akademik;
Database changed
MariaDB [akademik]>
```

*Gambar 1.2 Perintah Use*

### 4. Membuat Tabel

Perintah yang digunakan untuk membuat tabel dengan format sebagai berikut :

**CREATE TABLE t (a1 d1, a2 d2, a3 d3, ..., dan dn) ;**

Di mana :

t = nama tabel

a1 = nama atribut      d1 = domain atribut a1

Sebelum membuat tabel, maka struktur tabel harus diketahui terlebih dahulu. Type data yang ada pada MySQL/MariaDB.

Type Data	Keterangan
Char	Data karakter / string dengan Panjang tetap <= 5 digit
Varchar	Data karakter / string dengan Panjang tidak tetap >5 digit
Numeric	Data numerik real
Dec	Data numerik yang mengandung pecahan
Integer	Data bilangan bulat 2 byte
SmallInt	Data bilangan bulat 1 byte
Float	Data bilangan real
Double	Bilangan real dengan ketelitian tinggi untuk float
Blob	Data biner untuk menyimpan gambar atau suara
Date	Data tanggal (tahun-bulan-tanggal)
Datetime	Data tanggal dan waktu
Serial atau Auto Increment	Data numeric naik satu-satu otomatis
Boolean	Data nalar / logika

Nama tabel sebaiknya menggunakan huruf kecil, memiliki makna dan tanpa spasi, jika terdiri dari dua suku kata dapat menggunakan spasi, begitu juga untuk penamaan field. Kita akan membuat tabel mahasiswa di mana struktur dari tabel mahasiswa adalah sebagai berikut:

**Tabel mahasiswa**

<b>Nama field</b>	<b>Type</b>	<b>Panjang</b>	<b>Ket</b>
nrp	varchar	10	Primary
nama_mhs	varchar	30	
alamat	varchar	30	
tgl_lahir	date		

Perintah untuk membuat tabel mahasiswa adalah :

```
CREATE TABLE mahasiswa (nrp varchar(10) primary key,
nama_mhs varchar(30), alamat varchar(30), tgl_lahir
date) ;
```

Jika perintah yang diberikan tidak mengandung kesalahan maka akan tampil :

```
Query OK, 0 rows affected (... sec)
```

Kita dapat melihat tabel yang telah dibuat dengan memberikan perintah :

```
SHOW tables;
```

Tables_in_akademik
mahasiswa

1 row in set (...sec)

Untuk melihat kembali struktur tabel yang telah Anda buat, berikan perintah :

```
Desc mahasiswa;
```

Maka akan tampil sebagai berikut :

Field	Type	Null	Key	Default	Extra
nrp	varchar(10)	NO	PRI	NULL	
nama_mhs	varchar(30)	YES		NULL	
alamat	varchar(30)	YES		NULL	
tgl_lahir	date	YES		NULL	

## 5. Tugas 1.1

Lanjutkan pembuatan tabel untuk tabel-tabel yang ada dibawah ini :

**Tabel dosen**

<b>Nama field</b>	<b>Type</b>	<b>Panjang</b>	<b>Ket</b>
kode_dos	varchar	10	Primary
nama_dos	varchar	30	
alamat	varchar	30	
pend_terakhir	char	2	



**Tabel matakuliah**

<b>Nama field</b>	<b>Type</b>	<b>Panjang</b>	<b>Ket</b>
kode_kul	varchar	6	Primary
nama_kul	varchar	20	
sks	SmallInt		
semester	SmallInt		

**Tabel KRSHeader**

<b>Nama field</b>	<b>Type</b>	<b>Panjang</b>	<b>Ket</b>
kode_krs	char	6	Primary
semester	char	1	
tahun_akademik	char	4	

**Tabel KRSRinci**

<b>Nama field</b>	<b>Type</b>	<b>Panjang</b>	<b>Ket</b>
kode_rinci	serial	autoincrement	Primary
nama_krs	char	6	
nrp	varchar	9	
kode_kul	varchar	6	
nilai	char	2	

**Tabel jadwal**

<b>Nama field</b>	<b>Type</b>	<b>Panjang</b>	<b>Ket</b>
kode_jdw	serial	autoincrement	Primary
kode_dos	varchar	10	
kode_kul	varchar	6	
hari	char	26	
jam	char	11	
ruang	varchar	6	

## **6. Mengisi Data Dalam Tabel**

Perintah yang digunakan untuk mengisi data ke dalam tabel yang telah dibentuk adalah :

```
INSERT INTO mahasiswa  
VALUE ("1151800099", "Erwin Darmawan", "Puspiptek  
blok ... Serpong", "1985-02-06");
```

Dengan perintah diatas, maka akan dimasukan nilai (value) kedalam tabel secara berurutan, oleh karena itu pastikan nilai dan urutan field sesuai.

Jika ingin memasukan data lebih dari satu baris, perintahnya :

```
INSERT INTO mahasiswa  
VALUE
```

```
("1151800067", "Erwin Guntara", "Puspiptek blok ...  
Serpong", "1989-02-06"),  
("1151800076", "Erwin Aska", "Puspiptek blok ...  
Serpong", "1999-05-06");
```

Perintah INSERT... VALUES diatas cocok untuk tabel dimana kita telah mengetahui urutan dari kolom yang ada. Namun dalam situasi kita tidak mengetahui urutan kolom dari tabel, kita harus mendefenisikan urutan kolom yang akan diisi dengan cara penulisan query INSERT... VALUES.

Untuk situasi dimana kolom yang akan diisi tidak diketahui urutannya, atau kita hanya akan mengisi sebagian kolom saja, maka kita harus mendefinisikan kolom-kolom mana saja yang akan digunakan. Untuk keperluan tersebut, MySQL menyediakan variasi query INSERT, yaitu:

```
INSERT INTO nama_tabel (kolom1,kolom2,...) VALUES  
(nilai_kolom1,nilai_kolom2,...);
```

kolom1 adalah nama kolom yang akan kita input dengan nilai\_kolom1, dan kolom2 adalah nama kolom yang akan diisi dengan data pada nilai\_kolom2. Urutan nama kolom dengan nilai yang akan diisi harus sesuai.

Berikut contoh dengan mendefinisikan kolom:

```
INSERT INTO mahasiswa (nrp,nama_mhs,alamat,tgl_lahir)  
VALUES ("115190023","Sutan Batubara","Muncul","1992-  
08-02);
```

## 7. Menampilkan Data dari Tabel

Untuk melihat isi tabel mahasiswa yang telah diisi berikan perintah :

```
SELECT nrp, nama_mhs, alamat, tgl_lahir  
FROM mahasiswa;
```

Contoh output :

nrp	nama_mhs	alamat	tgl_lahir
1151800099	Erwin Darmawan	Puspiptek blok ... Serpong	1985-02-06

Setelah perintah **SELECT** diikuti oleh nama kolom yang akan ditampilkan, untuk query performance sebaiknya dituliskan. Untuk menampilkan semua data dapat menggunakan perintah dibawah ini, dengan mengganti nama kolom menjadi \* :

```
SELECT *  
FROM mahasiswa;
```

## 8. Pengubahan Struktur Tabel

Pengubahan struktur tabel dapat berupa panambahan field, penghapusan field dan pengaturan lebar field yang sudah dibuat pada tabel.

### Menambahkan field dalam tabel :

Jika pada tabel mahasiswa ingin ditambahkan atribut kelamin char (1), perintah yang digunakan adalah :

```
ALTER TABLE mahasiswa add kelamin char(1);
```

Untuk melihat apakah field tersebut ditambahkan di tabel mahasiswa, berikan perintah :

```
DESC mahasiswa;
```

Field	Type	Null	Key	Default	Extra
nrp	varchar(10)	NO	PRI	NULL	
nama_mhs	varchar(30)	YES		NULL	
alamat	varchar(30)	YES		NULL	
tgl_lahir	date	YES		NULL	
kelamin	char(1)	YES		NULL	

### Menghapus field dalam tabel

Jika terdapat field yang tidak diperlukan dalam sebuah tabel dapat dihapus dengan perintah :

```
ALTER TABLE mahasiswa DROP kelamin;
```

### Mengubah lebar field dalam tabel

Jika terdapat field yang tidak sesuai dan ingin diubah, baik nama maupun ukurannya dapat menggunakan perintah :

```
ALTER TABLE mahasiswa CHANGE nama_mhs nama  
varchar(30);
```

## 9. Melihat Struktur Tabel

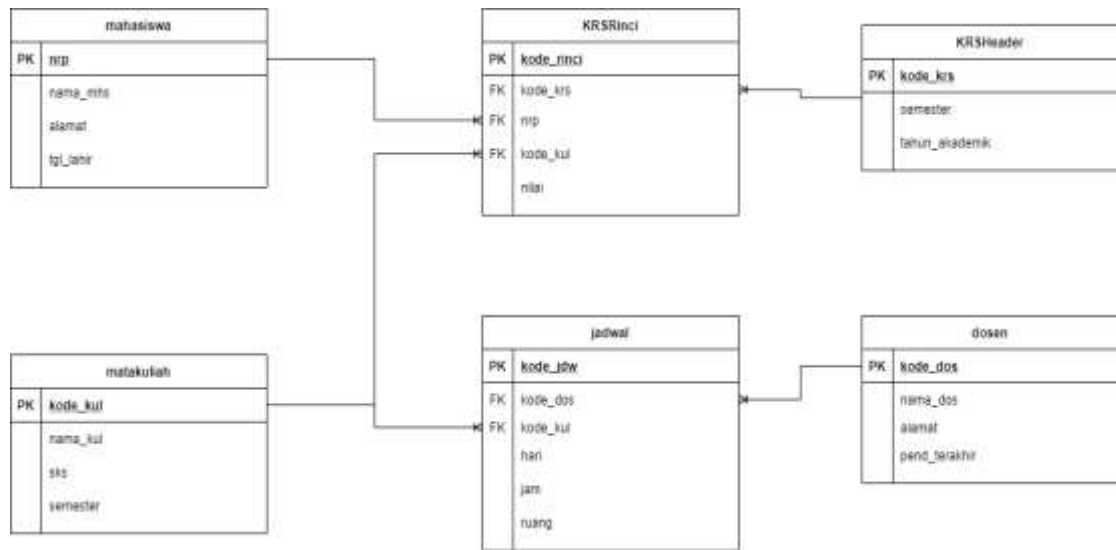
Untuk melihat struktur tabel yang telah dibuat dapat menggunakan perintah :

```
DESC matakuliah;
```

Field	Type	Null	Key	Default	Extra
kode_kul	varchar(6)	NO	PRI	NULL	
nama_kul	varchar(20)	YES		NULL	
sks	smallint(6)	YES		NULL	
semester	smallint(6)	YES		NULL	

## 10. ERD (Entity Relationship Diagram)

Setelah berhasil membuat tabel-tabel diatas, kita berhasil membuat database yang berelasi, berikut adalah diagram dari database :



Gambar 1.3 ERD Database

Pada beberapa tool editor database seperti phpMyAdmin, workbench dan lainnya, relasi database ini sudah otomatis dibuat menyesuaikan dengan tabel yang berada dalam database. Oleh karena itu bisa lebih mudah. Beberapa aplikasi ada yang menjembatani dari membuat ERD kemudian akan mengenerate tabel.

## 11. Menghapus Tabel

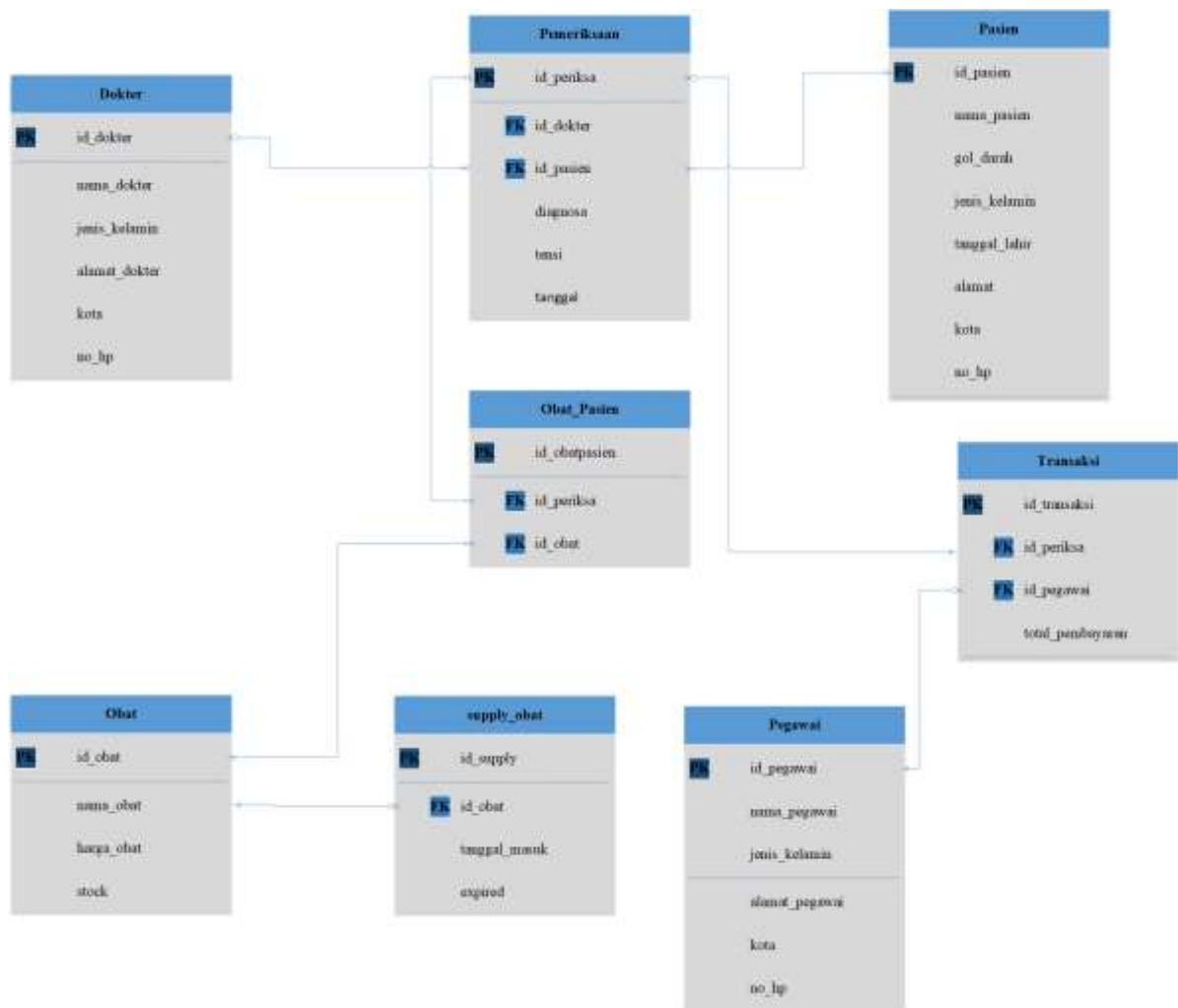
Untuk menghapus tabel dapat menggunakan perintah :

**DROP TABLE namaTabel;**

Mohon berhati-hati dalam menggunakan perintah ini, karena kesalahan dalam drop tabel maka tabel akan terhapus.

## 12. Tugas 1.2

Buatlah database klinik yang sesuai dengan ERD sebagaimana ditunjukan pada gambar 1.4.



Gambar 1.4 ERD Tugas Database

## MODUL 2

### Tipe Data, Operator Aritmatika

#### A. Tujuan Belajar

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami konsep aritmatika pada MySQL/MariaDB
2. Memahami cara melakukan operasi aritmatika di MySQL/MariaDB tanpa tabel
3. Memahami cara melakukan operasi aritmatika di MySQL/MariaDB pada field tabel

#### B. Praktikum

##### 1. Operator Aritmatika

Ekspresi dapat digunakan untuk memperoleh suatu nilai dari hasil perhitungan. Misalkan untuk memperoleh suatu hasil perhitungan penjualan barang maka digunakan rumus berikut :

Bayar = jmlBeli \* harga

Operator \* digunakan untuk mengalikan jmlBeli dan harga. Operator aritmatika mempunyai tingkatan yang berbeda seperti yang digunakan dalam pemrograman.

Operator	Fungsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Sisa hasil bagi

##### 2. Penggunaan Operator Aritmatika Tanpa Tabel

Dalam MySQL operator aritmatika dapat digunakan dengan perintah SELECT seperti berikut :

```
SELECT 7 + 3,
       3.00*2.50,
       "10 jari" - "5 jari";
```

Perintah ini akan diterjemahkan oleh MySQL sebagai perintah untuk menampilkan tabel sebagai berikut :

7+3	3.00 * 2.50	"10 jari" – "5 jari"
10	7.50	5

1 row in set, 2 warnings (0.003 sec)

Dari hasil diatas, terlihat karakter yang mengandung angka yaitu “10 jari” – “5 jari” diterjemahkan sebagai salah satu nilai dengan hasil 5.

Perhatikan contoh berikut :

```
SELECT "meja" + "kursi",
      -9 * 10;
```

"meja" + "kursi"	-9 * 10
0	-90

1 row in set, 2 warnings (0.000 sec)

Ternyata MySQL tidak bisa menambahkan data karakter “meja” + “kursi”. Operator aritmatika tidak berlaku untuk data karakter.

Contoh lain:

```
SELECT 20 + 30,
      10 - 9,
      300 % 24;
```

20 + 30	10-9	300 % 24
50	1	12

1 row in set (0.000 sec)

### 3. Penggunaan Operator Aritmatika Dalam Tabel

Operator aritmatika dapat juga digunakan dalam tabel dengan menambahkan atribut turunan. Buat sebuah database dengan nama penggajian, kemudian buat tabel gaji dengan atribut **nama varchar(20)**, **gajipokok int(8)**, **tunjangan int(8)**, **kehadiran smallint**.

```
CREATE DATABASE penggajian; [enter]
Query OK, 1 row affected (0.003 sec)
```

Ketik USE penggajian untuk membuka database penggajian dan operasi / perintah selanjutnya ada di dalam database penggajian.

```
USE penggajian; [enter]
Database changed
```

Database penggajian siap untuk digunakan. Sekarang berikan perintah untuk membuat tabel.

```
CREATE TABLE gaji (nama varchar(30), gajipokok int(8),
tunjangan int(8), kehadiran smallint); [enter]
Query OK, 0 rows affected (0.031 sec)
```

Perintah ini menyebabkan terbentuk tabel gaji dengan field-field yang disebutkan diatas, untuk melihat deskripsi tabel gaji berikan perintah berikut :

**DESC gaji;**

Field	Type	Null	Key	Default	Extra
nama	varchar(30)	YES		NULL	
gajipokok	int(8)	YES		NULL	
tunjangan	int(8)	YES		NULL	
kehadiran	smallint(6)	YES		NULL	

4 rows in set (0.028 sec)

Tipe data integer dapat ditentukan panjangnya, misalkan int(8), tetapi jika tidak diisi maka akan dimasukkan default int (11), sedangkan untuk smallint jika tidak diisi otomatis akan diisi smallint(6) oleh MySQL.

Lakukan pengisian data terhadap tabel gaji yang telah dibuat dengan memberikan perintah berikut satu persatu :

```
INSERT INTO gaji VALUE ("Ali", 5000000, 2500000, 25);
INSERT INTO gaji VALUE ("Nina", 5000000, 2000000, 24);
INSERT INTO gaji VALUE ("Budi", 4500000, 2000000, 25);
```

Untuk melihat isi tabel gaji dapat menggunakan perintah berikut :

**SELECT \* FROM gaji;**

nama	gajipokok	tunjangan	kehadiran
Ali	5000000	2500000	25
Nina	5000000	2000000	24
Budi	4500000	2000000	25

3 rows in set (0.000 sec)

Selanjutkan tambahkan field transport INT, gajibruto INT, pph INT, gajinet INT pada tabel gaji. Ketik perintah berikut :



```
ALTER TABLE gaji ADD transport int,
ADD gajibruto int,
ADD pph int,
ADD gajinet int;
```

Untuk melihat struktur tabel gaji yang telah diubah, ketik perintah berikut :

```
DESC gaji;
```

Field	Type	Null	Key	Default	Extra
nama	varchar(30)	YES		NULL	
gajipokok	int(8)	YES		NULL	
tunjangan	int(8)	YES		NULL	
kehadiran	smallint(6)	YES		NULL	
transport	int(11)	YES			
gajibruto	int(11)	YES			
pph	int(11)	YES			
gajinet	int(11)	YES			

8 rows in set (0.035 sec)

Semua data dengan tipe integer yang tidak ditentukan panjangnya akan diisi INT(11) secara otomatis.

Untuk melihat isi tabel gaji yang baru, menggunakan perintah berikut :

```
SELECT * FROM gaji;
```

nama	gajipokok	tunjangan	kehadiran	transport	gajibruto	pph	gajinet
Ali	5000000	2500000	25	NULL	NULL	NULL	NULL
Nina	5000000	2000000	24	NULL	NULL	NULL	NULL
Budi	4500000	2000000	25	NULL	NULL	NULL	NULL

3 rows in set (0.001 sec)

Untuk mengubah isi field dari NULL menjadi suatu nilai pada field transport, gajibruto, pph dan gajinet dapat menggunakan perintah berikut :

```

UPDATE gaji SET
transport = kehadiran * 15000,
gajibruto = gajipokok + tunjangan + transport,
pph = (gajipokok + tunjangan) * 15 / 100,
gajinet = gajibruto-pph;

```

Berikan perintah berikut untuk melihat isi tabel gaji sekarang :

```

SELECT * FROM gaji;

```

nama	gajipokok	tunjangan	kehadiran	transport	gajibruto	pph	gajinet
Ali	5000000	2500000	25	375000	7875000	1125000	6750000
Nina	5000000	2000000	24	360000	7360000	1050000	6310000
Budi	4500000	2000000	25	375000	6875000	975000	5900000

## MODUL 3

### Membuat dan Memanipulasi Tabel

#### CREATE, ALTER, DROP

---

#### A. Tujuan Belajar

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami cara manipulasi tabel pada MySQL/MariaDB
2. Memahami Constraint pada MySQL/MariaDB

#### B. Praktikum

##### 1. Membuat Tabel

Tabel merupakan dasar untuk membentuk suatu database relasional. Tabel-tabel harus mempunyai relasi satu dengan yang lainnya. Tabel-tabel akan terkumpul dalam suatu database tertentu. Jika belum ada database maka harus dibuat dulu databasenya dengan perintah “CREATE DATABASE namaDatabase”, jika sudah ada database tinggal menggunakan database dengan menggunakan perintah “USE namaDatabase”.

Contoh :

```
CREATE DATABASE rentalvcd;
```

Agar database rentalvcd dapat digunakan harus diaktifkan terlebih dahulu dengan memberikan perintah berikut :

```
USE rentalvcd;
```

Database rentalvcd dapat digunakan, selanjutnya akan diisi dengan tabel-tabel yang dibutuhkan untuk menyimpan data-data yang diperlukan dengan menggunakan perintah berikut, setiap perintah (setelah titik koma ;) tekan [enter] :

```
CREATE TABLE vcd
(kode_vcd char(5) PRIMARY KEY,
judul varchar(30) not null,
kategori varchar(15) not null,
produksi varchar(20) not null,
tahun char(4),
pemeran_utama varchar(20) not null,
harga_sewa int);
```

```
CREATE TABLE pelanggan
(kode_pel char(5) PRIMARY KEY,
nama varchar(30) not null,
kelamin char(1) not null,
alamat varchar(30) not null,
telepon varchar(15),
noKTP varchar(20));
```

```
CREATE TABLE pinjamheader
(kode_pinj char(5) PRIMARY KEY,
tgl_pinj date not null,
jml_pinjam smallint(3));
```

```
CREATE TABLE pinjamrinci
(kode_rinci char(5) PRIMARY KEY,
kode_pinj char(5),
kode_pel char(5) not null,
kode_vcd char(5) not null);
```

Untuk melihat deskripsi dari tabel yang telah dibuat dapat menggunakan perintah :

```
DESC vcd;
```

Field	Type	Null	Key	Default	Extra
kode_vcd	char(5)	NO	PRI	NULL	
judul	varchar(30)	NO		NULL	
kategori	varchar(15)	NO		NULL	
produksi	varchar(20)	NO		NULL	
tahun	char(4)	YES		NULL	
pemeran_utama	varchar(20)	NO		NULL	
harga_sewa	int(11)	YES		NULL	

7 rows in set (0.025 sec)

### TUGAS 3.1

Tampilan deskripsi dari tabel berikut :

- Tabel pelanggan
- Tabel pinjamheader
- Tabel pinjamrinci

Perhatikan hasil yang ditampilkan layer !

## 2. Constraint

*Constraint* digunakan untuk membatasi suatu kolom agar tidak bisa mengandung nilai tertentu, atau hanya dapat menyimpan nilai tertentu saja. Ada beberapa constraint yang sering digunakan, yaitu not null, primary key.

*Not null* : Digunakan untuk kolom yang di atur sebagai not null harus di isi (tidak boleh kosong).

*Primary Key* : Digunakan untuk membatasi agar sebuah kolom tidak mengandung dua baris yang memiliki nilai yang sama (harus unik) dan kolom tersebut tidak boleh kosong.

### 3. Manipulasi Tabel

Tabel-tabel yang telah dibuat dapat dirubah strukturnya dengan cara :

- a. Menambahkan / mengurangi field / kolom
- b. Mengubah definisi kolom
- c. Mengganti nama kolom atau tabel
- d. Menambnah atau mengurangi constraint

Perubahan-perubahan itu dapat didukung oleh perintah ALTER TABLE

#### 3.1. Menambahkan kolom baru → alter + add

Perintah yang digunakan :

```
ALTER TABLE nama_tabel  
ADD nama_kolom type_kolom,  
...;
```

Tambah kolom pekerjaan dan agama pada tabel pelanggan. Berikan perintah berikut :

```
ALTER TABLE pelanggan  
ADD pekerjaan varchar(15) ,  
ADD agama varchar(10) ,  
ADD alamat_kantor varchar(20) ;
```

Untuk melihat hasil penambahan kolom ini jangan lupa berikan perintah “DESC pelanggan;”.

#### 3.2. Menghapus kolom → alter + drop

Jika ada kolom yang akan dihilangkan dari tabel yang telah dibuat dapat dilakukan dengan mudah. Pada tabel pelanggan kolom alamat\_kantor akan dihilangkan, maka dapat menggunakan perintah berikut :

```
ALTER TABLE pelanggan  
DROP alamat_kantor;
```

Jika dilihat pada tabel pelanggan sekarang sudah tidak ditemukan lagi ada kolom alamat\_kantor.

#### 3.3. Memodifikasi Kolom → alter + change

Andaikan tabel yang sudah dibuat ada kolom yang perlu dimodifikasi, hal ini dimungkinkan dalam MySQL. Pada tabel pelanggan akan diubah tipe data dari kolom agama varchar(10) menjadi agama char(1). Berikan perintah berikut untuk mengubahnya :

```
ALTER TABLE pelanggan  
CHANGE agama agama char(1) ;
```

Perintah CHANGE dapat diganti dengan perintah MODIFY untuk memberikan hasil yang sama, berikut perintahnya :

```
ALTER TABLE pelanggan  
MODIFY agama char(1);
```

### TUGAS 3.2

Buatlah tabel baru yang terdiri dari 2 kolom yaitu *kode\_agama char(1)* dan *nama\_agama varchar(10)* untuk menyimpan data kode dan nama agama yang diakui di Indonesia.

Kemudian ubah kolom agama yang ada pada tabel pelanggan menjadi kode agama *char(1)*. Lalu ubah lagi kolom kode\_agama menjadi *enum ('1', '2', '3', '4', '5')*.

### 3.4. Melihat Daftar Database dan Tabel

Untuk mengetahui daftar database yang ada dapat menggunakan perintah berikut :

```
SHOW databases;
```

Maka dilayar akan ditampilkan semua database yang telah dibuat sebelumnya. Tampilannya seperti berikut :

Database
akademik
information_schema
mysql
penggajian
performance_schema
phpmyadmin
rentalvcd

7 rows in set (0.001 sec)

Database yang aktif adalah database rentalvcd, sehingga kalau kita ingin melihat daftar tabel, maka yang akan ditampilkan adalah semua tabel yang ada pada database rentalvcd. Berikut adalah perintah dan tampilannya :

```
SHOW tables;
```

Tables_in_rentalvcd
pelanggan
pinjamheader

pinjamrinci
vcd

4 rows in set (0.001 sec)

Untuk melihat tabel pada database yang lain, kita harus mengaktifkan dahulu database nya. Misalnya kita jika menampilkan tabel-tabel yang ada pada database penggajian, maka harus perlu menggunakan perintah berikut :

**USE penggajian;**

Setelah database aktif, berikan perintah berikut :

**SHOW tables;**

Tables_in_penggajian
Gaji

1 row in set (0.001 sec)

### **TUGAS 3.3**

1. Tampilkan struktur tabel dari tabel-tabel yang ada di database akademik
2. Aktifkan database akademik dan tampilkan tabel-tabel yang ada
3. Tampilkan struktur tabel-tabel yang ada pada database akademik

## MODUL 4

### Memanipulasi Data

#### INSERT, UPDATE, DELETE

---

#### A. Tujuan Pembelajaran

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami cara memanipulasi data tabel pada MySQL

#### B. Praktikum

##### 1. Menambah Data

Perintah yang digunakan untuk mengisi tabel-tabel yang telah dibuat adalah INSERT INTO VALUE. Tabel-tabel tidak dapat diisi sebelum databasenya diaktifkan. Ketik perintah berikut untuk mengaktifkan database akademik yang sudah dibuat pada modul 1.

```
USE akademik;
```

Agar kita tidak salah dalam menuliskan nama tabel, sebaiknya dilihat kembali tabel-tabel yang ada dalam database akademik menggunakan perintah berikut :

```
SHOW tables;
```

Perhatikan tampilan yang ada di layer monitor.

Sebelum mengisi tabel dengan data, kita harus tahu struktur tabel yang akan diisi menggunakan perintah berikut :

```
DESC mahasiswa;
```

Field	Type	Null	Key	Default	Extra
nrp	varchar(10)	NO	PRI	NULL	
nama_mhs	varchar(30)	YES		NULL	
alamat	varchar(30)	YES		NULL	
tgl_lahir	date	YES		NULL	
kelamin	char(1)	YES		NULL	

Kita akan mengisi tabel mahasiswa dengan data menggunakan perintah berikut :

```
INSERT INTO mahasiswa
VALUES ("1150500002", "Ibnu Pratama", "Jakarta",
"1985-03-15", "P"),
( "1150500003", "Prapaska", "Jl. Lumbu Timur IIE
Blok VI Bekasi", "1988-01-14", "P"),
( "1150500005", "Herni Adriyani", "Permata
Pamulang, Jl Betet V", "1987-10-17", "P"),
```



```
("1150500018", "Denny Setiawan", "Jl H Mawi  
Parung-Bogor", "1987-06-22", "L");
```

Jika tidak ada kesalahan maka di layer akan menampilkan informasi seperti berikut :

*Query OK, 4 rows affected, 1 warning (0.005 sec)*

*Records: 4 Duplicates: 0 Warnings: 1*

Untuk melihat hasil pemasukan data, gunakan perintah berikut :

```
SELECT * FROM mahasiswa;
```

nrp	nama	alamat	tgl_lahir	kelamin
1150500002	Ibnu Pratama	Jakarta	1985-03-15	P
1150500003	Prapaska	Jl. Lumbu Timur IIE Blok VI Be	1988-01-14	P
1150500005	Herni Adriyani	Permata Pamulang, Jl Betet V	1987-10-17	P
1150500018	Denny Setiawan	Jl H Mawi Parung-Bogor	1987-06-22	L

## 2. Tugas 4.1

Isi tabel yang ada didalam database akademik sesuai dengan data berikut :

### Tabel matakuliah

Kode_kul	Nama_kul	Sks	Semester
IF-310	Database	3	5
IF-312	Sistem Operasi	3	5
IF-313	Ansis	3	5

### Tabel dosen

kode_dos	nama_dos	alamat	pend_terakhir
DT001	Sumiarti, Ir. M.Kom	Batan Indah N-33 Serpong	S-2
TT005	Budi Prasetyo drs. MT	Komplek Batan Cisauk	S-2
DT004	Yustina Sri S. ST. MT	Pondok Cabe	S-2

**Tabel KRSheader**

Kode_krs	Semester	Tahun_akademik
IF1001	Ganjil	2008/2009
IF1002	Ganjil	2008/2009
IF1003	Ganjil	2008/2009

**Tabel KRSrinci**

Kode_rinci	Kode_krs	NRP	Kode_kul	Nilai
1	IF1001	1150500003	IF-310	B
2	IF1001	1150500003	IF-312	A
3	IF1001	1150500003	IF-313	B
4	IF1002	1150500005	IF-310	B
5	IF1002	1150500005	IF-312	A
6	IF1003	1150500018	IF-310	A
7	IF1003	1150500018	IF-312	A
8	IF1003	1150500018	IF-313	A

**Tabel jadwal**

Kode_jdw	kode_dos	Kode_kul	Hari	Jam	Ruang
1	DT001	IF-310	Senin	08.00 – 10.30	D-7
2	DT001	IF-310	Jumat	08.00 – 10.30	D-7
3	TT005	IF-313	Kamis	10.40 – 13.10	D-7
4	TT005	IF-313	Senin	10.40 – 13.10	D-6
5	DT004	IF-312	Selasa	08.00 – 10.30	D-3

### 3. Mengubah Data

Jika ada data yang telah dimasukkan ke dalam tabel ada yang salah, data tersebut bisa dibetulkan dengan menggunakan perintah UPDATE. Bentuk perintah ini adalah :

```
UPDATE namaTabel
    SET kolom_1 = nilai_baru_1,
        kolom_2 = nilai_baru_2,
        ...
        kolom_n = nilai_baru_n
WHERE kondisi;
```

Pada tabel mahasiswa diatas data alamat dan tgl\_lahir Ibnu Pratama salah, untuk memperbaikinya harus diisi dengan data yang benar. Perintah yang diberikan adalah :

```
UPDATE mahasiswa
SET alamat = "Jl Mangga Besar, Jakarta Pusat",
    tgl_lahir = "1987-7-17"
WHERE nrp = "1150500002";
```

Jika Anda ingin mengubah beberapa data sekaligus, bisa menggunakan perintah :

```
UPDATE nama_tabel
SET kolom_2 = "nilai_baru"
WHERE kolom_1 = "nilai_1" OR
    kolom_1 = "nilai_2";
```

Tampilkan kembali tabel mahassiswa yang telah diisi. Ada kesalahan kelamin untuk 2 mahasiswa, yaitu NRP 1150500002 dan 115505500003 yang seharusnya adalah "L". Perbaiki data tersebut dengan menggunakan perintah :

```
UPDATE mahasiswa
SET kelamin = "L"
WHERE nrp = "1150500002" OR
    nrp = "1150500003";
```

Tampilkan kembali tabel mahasiswa, perhatikan data yang telah diubah.

#### 4. Menghapus Data

Perintah yang dapat digunakan untuk menghapus adalah **DELETE** dengan format penulisan sebagai berikut :

```
DELETE FROM namaTabel
WHERE kondisi;
```

Perintah **WHERE** bisa digunakan jika ingin menghapus baris tertentu yang tidak diperlukan

##### 4.1.Menghapus sebuah baris

Untuk menghapus baris yang berisi data mahasiswa yang mempunyai NRP 1150500002 dapat menggunakan perintah sebagai berikut :

```
DELETE FROM mahasiswa
WHERE nrp = "1150500002";
```

Tampilkan kembali tabel mahasiswa, perhatikan isi tabel sekarang, data mahasiswa dengan NRP 1150500002 sudah terhapus dari tabel mahasiswa.

nrp	nama	alamat	tgl_lahir	kelamin
1150500003	Prapaska	Jl. Lumbu Timur IIE Blok VI Be	1988-01-14	L

1150500005	Herni Adriyani	Permata Pamulang, Jl Betet V	1987-10-17	P
1150500018	Denny Setiawan	Jl H Mawi Parung-Bogor	1987-06-22	L

#### 5. Tugas 4.2

Tambahkan data untuk tabel mahasiswa dengan data lain yang Anda ketahui.

#### A. TUJUAN

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami 3 query dasar pada MySQL

#### B. Praktikum

##### 1. Query

Setelah dilakukan pengisian data ke dalam table-table yang telah dibuat, maka selanjutnya dapat dilakukan operasi pencarian data sesuai dengan kebutuhan. Proses pencarian data terhadap tabel-tabel yang terpilih sebagai tempat pencarian data sesuai dengan kriteria yang ditetapkan disebut dengan Query. Perintah dasar dalam query ada 3, yaitu **SELECT**, **FROM**, **WHERE**.

**SELECT** : Perintah ini digunakan untuk menetapkan daftar atribut (field) yang diinginkan sebagai hasil query.

**FROM** : Perintah ini digunakan untuk menetapkan tabel (gabungan tabel) yang akan ditelusuri selama query data dilakukan.

**WHERE** : Perintah ini digunakan sebagai kriteria yang harus dipenuhi dalam memperoleh hasil query

Bentuk umum dari ini adalah :

```
SELECT field1, field2, ... fieldn
FROM tabel1, tabel2, ..., tabeln
WHERE kondisi;
```

##### 2. Tugas 5.1

- a. Tambahkan data diri Anda dan 2 teman Anda ke dalam tabel mahasiswa
- b. Kemudian isi tabel matakuliah dengan data sebagai berikut :

Kode_kul	Nama_kul	Sks	Semester
DK9042	Analisa Algoritma I	2	3
DK9053	Struktur Data	3	3
IF9093	System Berkas	3	3
DK9063	Aljabar Linier & Matriks	3	3
IF9103	Sistem Digital	3	3
IF9113	Pemrograman Berorientasi Objek + P	3	3
DK9123	Matematika Diskrit II	3	3
IF9182	Komputer & Masyarakat	2	5

IF9223	Rekayasa Piranti Lunak	3	5
IF9233	Sistem Informasi	3	5

c. Rubah kode kuliah berikut :

Kode_kul		Nama_kul	
LAMA	BARU	LAMA	BARU
IF-310	IF9213	Database	Basisdata + P
IF-312	IF9203	Sistem Operasi	Sistem Operasi + P
IF-313	IF9173	Ansis	Analisa & Perancangan Sistem

d. Masukkan data KRS untuk 3 data tambahan (Anda + 2 teman Anda)

e. Tambahkan data dosen yang mengajar ke tabel Anda, kolom yang datanya tidak diketahui dapat dikosongkan.

Tugas ini harus diselesaikan dulu sebelum melanjutkan ke materi 3 !!!!!!!!!!!

### 3. Menampilkan Data Tanpa Syarat

Jika ingin menampilkan semua data yang ada di tabel dosen dapat menggunakan perintah :

```
SELECT *
FROM dosen;
```

Tanda \* mewakili semua atribut / field yang ada di tabel dosen. Contoh hasil yang didapatkan adalah :

kode_dos	nama_dos	alamat	pend_terakhir
DT001	Sumiarti, Ir. M.Kom	Batan Indah N-33 Serpong	S-2
TT005	Budi Prasetyo drs. MT	Komplek Batan Cisauk	S-2
DT004	Yustina Sri S. ST. MT	Pondok Cabe	S-2

Apabila Anda ingin hanya menampilkan field / kolom nama\_dos dan pend\_terakhir saja maka dapat menggunakan perintah berikut :

```
SELECT nama_dos, pend_terakhir
FROM dosen;
```

Hasil yang didapatkan adalah :

nama_dos	pend_terakhir
Sumiarti, Ir. M.Kom	S-2
Budi Prasetyo drs. MT	S-2

#### 4. Tugas 5.2

Buatlah perintah untuk menampilkan data-data berikut :

- NRP dan nama mahasiswa
- Semua kolom yang ada pada tabel data matakuliah
- Nama matakuliah dan sks

#### 5. Menampilkan Data Bersyarat

Perintah **WHERE** hanya digunakan jika data yang akan ditampilkan dipilih berdasarkan kondisi tertentu. Untuk menuliskan kondisi yang diinginkan perlu digunakan operator relasional (=, <, <=, >, >=, <>) yang membandingkan suatu nilai dengan nilai yang lain. Selain itu juga dapat digunakan operator logika (OR, AND, NOT, XOR).

Misal ingin menampilkan data matakuliah yang ada pada semester 5, dapat menggunakan perintah berikut :

```
SELECT *
FROM matakuliah
WHERE semester = 5;
```

Perhatikan data yang ditampilkan

Jika ingin menampilkan data NRP, nama\_mahasiswa yang mempunyai NRP > 1150500003

```
SELECT nrp, nama
FROM mahasiswa
WHERE NRP > "1150500003";
```

Tanda kutip yang mengapit angka 1150500003 digunakan karena type data NRP adalah teks (varchar), sedangkan untuk semester tipe datanya number, sehingga tidak perlu tanda kutip.

Untuk menampilkan data matakuliah yang ada di semester 5 dan mempunyai bobot 2 sks dapat menggunakan perintah berikut :

```
SELECT *
FROM matakuliah
WHERE semester = 5 AND sks = 2;
```

Tampilannya seperti berikut :

Kode_kul	Nama_kul	Sks	Semester
IF9182	Komputer & Masyarakat	2	5

Dalam tabel matakuliah yang telah diisi hanya ada 1 matakuliah yang memenuhi syarat yang diminta, yaitu semester = 5 dan sks = 2

## 6. Tugas 5.3

Tampilkan data dengan kriteria berikut :

- NRP dan nama mahasiswa perempuan
- Hari dan jam penggunaan ruang D-7

## 7. Query Beberapa Tabel

Query dapat dilakukan terhadap 2 atau lebih tabel yang mempunyai relasi.

Misalkan kita ingin menampilkan data matakuliah yang diajarkan dosen dapat menggunakan perintah berikut :

```
SELECT      dosen.nama_dos,      matakuliah.nama_kul,
matakuliah.sks
FROM dosen, matakuliah, jadwal
WHERE dosen.kode_dos = jadwal.kode_dos AND
      matakuliah.kode_kul = jadwal.kodekul;
```

Tampilannya sebagai berikut :

Dosen.nama_dos	Matakuliah.nama_kul	Sks
Sumiarti, Ir. M.Kom	Basisdata + P	3
Budi Prasetyo drs. MT	Analisa dan Perancangan Sistem	3
Yustina Sri S. ST. MT	Sistem Operasi	3
...	...	...

Jika tabel yang digunakan tidak mempunyai relasi, maka hasil yang ditampilkan akan menggabungkan antara tabel 1 dan tabel 2

Contoh :

```
SELECT      mahasiswa.nrp,      mahasiswa.nama,
matakuliah.nama_kul, matakuliah.sks
FROM mahasiswa, matakuliah;
```

Perhatikan hasil data yang ditampilkan!

## 8. Tugas 5.4

Tampilkan data berikut :

- Jadwal dosen (nama dosen, nama matakuliah, sks, hari, jam)
- Nama dosen, nama matakuliah, sks, hari, jam yang menggunakan ruang D-7



## MODUL 6

### Query Lanjutan dan Fungsi Agregat

---

#### A. Tujuan Pembelajaran

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami query lanjutan yang ada pada MySQL/MariaDB
2. Memahami fungsi agregat pada MySQL/MariaDB

#### B. Praktikum

##### 1. Operator BETWEEN

Operator ini digunakan untuk menangani operasi jangkauan. Misalkan ingin menampilkan data mahasiswa yang lahir antara tahun 1987 sampai 1990, dapat menggunakan perintah berikut :

```
SELECT nama, tgl_lahir
FROM mahasiswa
WHERE tgl_lahir BETWEEN "1987-01-01" AND "1990-01-01";
```

Perhatikan hasil yang ditampilkan!

##### 2. Operator IN

Operator ini berguna untuk melakukan pencocokan dengan salah satu yang ada pada suatu daftar nilai. Misalkan menampilkan nrp dan nama mahasiswa yang mempunyai NRP 1150500003, 1150500005, 1150500018, dapat menggunakan perintah berikut :

```
SELECT nrp, nama
FROM mahasiswa
WHERE nrp in ("1150500003", "1150500005", "1150500018");
```

Perhatikan hasil yang ditampilkan!

##### 3. Operator LIKE

Operator ini bermanfaat untuk mencari data yang mengandung 1 atau lebih karakter. Dalam melakukan pencarian perlu disebutkan wildcard seperti berikut :

- a. Tanda garis bawah (\_), berarti sebuah (1) karakter apa saja  
Contoh : a\_i akan sesuai dengan ani, ari, ali
- b. Tanda persen (%) berarti cocok dengan karakter apa saja dan berapapun panjangnya (termasuk karakter nol)  
Contoh : %a% cocok dengan apa saja yang mengandung karakter a atau A  
%a cocok untuk sesuatu yang berakhiran dengan a atau A  
a% cocok dengan sesuatu yang diawali dengan a atau A

Jika ingin menampilkan nama mahasiswa yang diawali dengan huruf d, dapat menggunakan perintah berikut :

```

SELECT nama
FROM mahasiswa
WHERE nama like "d%";

```

Perhatikan tampilannya !

Untuk menampilkan nrp dan nama mahasiswa angkatan 06 yang perempuan dapat menggunakan perintah berikut :

```

SELECT nama, nrp
FROM mahasiswa
WHERE nrp like "__06%" AND kelamin = "P";

```

Perhatikan underscore (\_) yang digunakan ada 3 buah, maksudnya nrp yang karakter 4 dan 5 berisi 06. Data yang dihasilkan adalah nama dan nrp angkatan 06 untuk semua jurusan. (Coba isi data mahasiswa jurusan lain angkatan 06, lalu lakukan perintah diatas sekali lagi, perhatikan hasilnya).

#### 4. Mengurutkan Data (ORDER BY)

Data yang disimpan didalam tabel belum tentu teratur. Saat menampilkan data, dapat ditampilkan dengan urutan yang diinginkan. Perintah untuk mengurutkan ada adalah ORDER BY. Pengurutan dapat dilakukan menaik (ascending) atau menurun (descending).

Untuk menampilkan data dosen teratur menurut nama menaik (ascending), dapat menggunakan perintah berikut :

```

SELECT *
FROM dosen
ORDER BY nama_dos;

```

Perhatikan tampilan yang dihasilkan!

Jika data mahasiswa yang ditampilkan ingin diurutkan dari yang paling muda, dapat menggunakan perintah berikut :

```

SELECT *
FROM mahasiswa
ORDER BY tgl_lahir desc;

```

Perhatikan data mahasiswa yang tampil, apakah sudah teratur ?

#### 5. Klausa Limit

Klausa limit digunakan untuk membatasi jumlah baris yang akan dihasilkan oleh suatu query.

```

SELECT nama_kul, sks
FROM matakuliah
LIMIT 5;

```

Perintah diatas membatasi hasil sebanyak 5.

## 6. Fungsi Agregat

Fungsi agregat adalah fungsi yang disediakan untuk menghasilkan sebuah nilai berdasarkan sejumlah data.

Fungsi	Keterangan
SUM()	Menghitung penjumlahan data
COUNT()	Menghitung banyak data
MAX()	Menghitung nilai terbesar
MIN()	Menghitung nilai terkecil
STD()	Menghitung standar deviasi
AVG()	Menghitung nilai rata-rata

Jika ingin menampilkan jumlah sks matakuliah yang ada di semester 5 dapat menggunakan perintah berikut :

```
SELECT SUM(sks)
FROM matakuliah
WHERE semester = 5;
```

Periksa jumlah sks matakuliah yang ada di semester 5 dan bandingkan dengan hasil yang ditampilkan.

Untuk menampilkan banyak matakuliah 3 sks di semester 5 dapat menggunakan perintah berikut :

```
SELECT COUNT(sks)
FROM matakuliah
WHERE sks = 3 AND semester = 5;
```

Perhatikan tampilan yang dihasilkan!

Untuk menampilkan nrp dan mahasiswa perempuan dapat menggunakan perintah berikut :

```
SELECT nrp, nama, kelamin, COUNT(nrp)
FROM mahasiswa
WHERE kelamin = "P";
```

## 7. Tugas 6.1

- Tampilkan nrp dan nama mahasiswa berjenis kelamin laki-laki dan hitung ada berapa banyak
- Tampilkan nrp dan nama mahasiswa peserta mata kuliah bnaissdata dan hitung jumlahnya

## 8. Mengelompokkan Data (GROUP BY)

Pengelompokan data sering digunakan bersamaan dengan fungsi count() ataupun sum(). Gunanya untuk mengelompokkan data sesuai dengan kelompok tertentu. Jika ingin menampilkan nama matakuliah, sks berdasarkan semester dapat menggunakan perintah berikut :

```
SELECT nama_kul, sks
FROM matakuliah
GROUP BY semester;
```

Perhatikan hasil yang ditampilkan

Untuk menampilkan nrp dan nama mahasiswa dikelompokkan berdasarkan jenis kelamin dapat menggunakan perintah berikut :

```
SELECT nrp, nama, kelamin
FROM mahasiswa
GROUP BY kelamin;
```

Perhatikan hasil yang ditampilkan

Perintah berikut digunakan untuk menampilkan banyak peserta per matakuliah:

```
SELECT matakuliah.nama_matkul, COUNT(*)
FROM matakuliah, krsrinci
WHERE matakuliah.kode_kul = krsrinci.kode_kul
GROUP BY krsrinci.kode_kul;
```

Klausula HAVING dapat digunakan untuk menentukan kondisi bagi GROUP BY, misalkan :

```
SELECT matakuliah.nama_matkul, COUNT(*)
FROM matakuliah, krsrinci
WHERE matakuliah.kode_kul = krsrinci.kode_kul
GROUP BY krsrinci.kode_kul
HAVING semester = 5;
```

Tampilan hanya menghasilkan matakuliah dan jumlah peserta untuk mata kuliah pada semester 5 saja. Bandingkan dengan hasil yang sebelumnya.

## 9. Tugas 6.2

- Tampilkan jumlah sks setiap semester yang harus diambil oleh mahasiswa
- Tampilkan banyaknya mata kuliah yang menggunakan ruangan D-7 setiap hari

### A. Tujuan Pembelajaran

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami fungsi-fungsi yang ada pada MySQL

### B. Praktikum

#### 1. Pendahuluan

Selain fungsi agregat yang telah dibahas pada modul 6. MySQL menyediakan beberapa fungsi yang bisa digunakan untuk melakukan berbagai operasi baik untuk melakukan query maupun untuk kepentingan manipulasi data. Fungsi tersebut antara lain adalah fungsi string, fungsi tanggal & waktu dan fungsi numeris.

#### 2. Fungsi String

Dalam MySQL banyak fungsi string yang disediakan diantaranya dapat dilihat pada tabel berikut :

Fungsi	Keterangan
ACS(str)	Menghasilkan nilai ASCII karakter terkiri pada string str, misal ASCII ("AB") → 65
BIN(n)	Menghasilkan string dari bilangan n, misal BIN(4) → "100"
BIT_LENGTH(str)	Menghasilkan jumlah bit dalam string str, misal BIT_LENGTH("A") → 8
CHAR(n, ...)	Menghasilkan string yang karakter secara individu bernilai n, misal CHAR(65, 66) → "AB"
LENGTH(str)	Menghasilkan jumlah karakter pada string str dengan ukuran byte, LENGTH("ABCDE") → 5
CONCAT(str, ...)	Menghasilkan string yang merupakan gabungan dari semua argumen / parameter nya
FORMAT(n, d)	Menghasilkan string dari format bilangan n dalam bentuk ###,###,##. D merupakan jumlah digit pecahan. Misal FORMAT(123456.5,2) → "123,456.50"
LEFT(str, n)	Menghasilkan string berupa n karakter terkiri dari string, misal LEFT("ABCDE", 2) → "AB"
MID(str, pos, n)	Menghasilkan string berupa n karakter pada str dimulai dari posisi pos, misal MID("ABCD",3,2) → "CD"
RIGHT(str, n)	Menghasilkan n buah karakter dari kanan

REPEAT(str)	Mengulang string str n kali, misal REPEAT("IF", 3) → "IFIFIF"
LOWER(str)	Mengubah string str menjadi huruf kecil
UPPER(str)	Mengubah string str menjadi huruf kapital

Jangan lupa setiap kali memulai praktikum harus melakukan perintah :

**USE akademik;**

Jika Anda ingin menggunakan database akademik

Lakukan penambahan data untuk tabel mahasiswa dengan memasukkan data berikut :

```
INSERT INTO mahasiswa
VALUES ("1150600009", "Vinda Martono", "Kampung
Cibogo, Kelapa Dua", "1988-08-30", "P"),
        ("1150600014", "Septian Pratama", "Jln H.Maih
Sawangan-Depok", "1988-01-02", "L"),
        ("1150600019", "Panji Setiadi", "Jln Pelopor",
"1987-12-12", "L");
```

Tampilkan semua data yang ada dalam tabel mahasiswa, kemudian jika ingin ditampilkan data mahasiswa angkatan 06 dapat menggunakan perintah berikut :

```
SELECT *
FROM mahasiswa
WHERE MID(nrp,4,2) = "06";
```

Perhatikan tampilan yang dihasilkan, bedakan dengan tampilan sebelumnya.

Untuk melakukan tampilan data yang mengandung pecahan dapat ditampilkan dalam format dengan pemisah ribuan dengan menggunakan perintah berikut :

```
SELECT FORMAT(123456.5,2) ;
```

Akan menampilkan hasil :

FORMAT(123456.5,2)
e123,456.50

Jika perintahnya diganti menjadi :

```
SELECT FORMAT(123456.5,2) as Bilangan;
```

Bilangan
123,456.50

Sekarang gunakan database penggajian, gunakan perintah berikut :

```
USE penggajian;
```

Gunakan perintah berikut :

```
SHOW tables;
```

Kemudian gunakan perintah berikut :

```
DESC gaji;
```

Hasil yang ditampilkan seperti berikut :

Field	Type	Null	Key	Default	Extra
nama	varchar(30)	YES		NULL	
gajipokok	int(8)	YES		NULL	
tunjangan	int(8)	YES		NULL	
kehadiran	smallint(6)	YES		NULL	

Gunakan perintah berikut :

```
SELECT nama, format(gajipokok, 0) as gapok  
FROM gaji;
```

Perhatikan sekarang tampilan gaji pokok sudah terformat dengan pemisah ribuan.

### 3. Tugas 7.1

Tampilkan nama, gajipokok, tunjangan, pph dan gajinet dengan format pemisah ribuan dengan 2 angka dibelakang koma.

### 4. Fungsi tanggal dan Waktu

MySQL juga menyediakan fungsi untuk memanipulasi data tanggal dan waktu, antara lain dapat dilihat pada tabel berikut :

Fungsi	Keterangan
ADDDATE(ekp, n)	n hari dari tanggal ekspresi, ADDDATE("2008-09-01", 5) → "2008/09/06"
DATEDIFF(ekp1, ekp2)	Menghasilkan selisih hari dari eks1 dan ekp2
CURDATE()	Tanggal sekarang
CURTIME()	Waktu sekarang
DAY(date)	Menampilkan tanggal (1 s/d 31) dari suatu tanggal, DAY("1945-08-17") → 17
hour(time)	Menghasilkan bagian jam dari suatu waktu, HOUR("12:30:05") → 12
MINUTE(time)	Menghasilkan bagian menit dari suatu waktu

MONTH(date)	Menghasilkan bagian bulan dari suatu tanggal berbentuk angka
MONTHNAME(date)	Menghasilkan nama bulan
TIME(ekp)	Menghasilkan jam, menit dan detik dari ekspresi tanggal dan jam
TIMEDIFF(ekp1, ekp2)	Menghasilkan selisih waktu antara kedua argument / parameter
YEAR(date)	Menghasilkan tahun dari suatu tanggal
DATE_FORMAT(tanggal, format)	Menghasilkan tanggal dengan format yang diatur. %c = bulan, %d = tanggal. %Y = tahun

Untuk menampilkan tanggal sekarang dapat menggunakan perintah berikut :

**SELECT CURDATE () ;**

CURDATE()
2021-03-05

Jika tampilan tanggal lahir diubah menjadi tanggal-bulan-tahun, dapat menggunakan perintah berikut :

**USE akademik;**  
**SELECT nrp, nama, DATE\_FORMAT(tgl\_lahir, "%d %M %Y")**  
**FROM mahasiswa;**

Perhatikan bulan lahir akan ditulis dengan nama bulan. Coba ganti %M menjadi %m, %Y menjadi %y, perhatikan tampilan yang baru. Perhatikan tabel format tanggal dan waktu berikut :

Penentu Format	Keterangan
%a	Nama hari disingkat dalam Bahasa inggris (Mon, Sat, dstnya)
%b	Nama bulan disingkat (Jan, Feb, dstnya)
%c atau %m	Kode bulan dalam numerik (1 s/d 12)
%D	Angka tanggal diikuti dengan akhiran 1 <sup>st</sup> , 2 <sup>nd</sup> , dstnya
%d	Tanggal dalam bentuk numerik (01 s/d 31)
%H	Jam (00 s/d 23)
%h	Jam (01 s/d 12)
%M	Nama bulan (Januari, Februari, dstnya)
%W	Nama hari (Sunday, Monday, dstnya)



%w	Kode hari (0 = minggu, 1 = senin, dstnya)
%Y	Tahun 4 digit
%y	Tahun 2 digit

Untuk menampilkan umur mahasiswa dapat menggunakan perintah berikut :

```
SELECT nrp, nama, DATEDIFF(CURDATE(), tgl_lahir)/365  
as umur  
FROM mahasiswa;
```

Perhatikan tampilan di layar

Umur yang ditampilkan mengandung pecahan

## 5. Fungsi Numerik

Selain fungsi-fungsi yang dijelaskan, MySQL juga mempunyai fungsi numerik antara lain seperti yang ada dalam tabel berikut :

<b>Fungsi</b>	<b>Keterangan</b>
ACOS(x)	Menghasilkan arccosine dari bilangan x
ASIN(x)	Menghasilkan arcsine dari bilangan x
ATAN(x)	Menghasilkan arctangent dari bilangan x
COS(x)	Menghasilkan nilai cos x, x dalam radian
SIN(x)	Menghasilkan nilai sin x, x dalam radian
TAN(x)	Menghasilkan nilai tan x, x dalam radian
DEGREES(x)	Menghasilkan nilai dalam derajat, x dalam radian
RADIAN(x)	Konversi dari derajat ke radian
CEIL(x)	Menghasilkan bilangan bulat terkecil yang tidak kurang dari x, CEIL(6,7)=7, CEIL(-6,7)=6
FLOOR(x)	Menghasilkan bilangan bulat terbesar yang tidak lebih dari x, FLOOR(6.7) = 6, FLOOR(-6.7)=-7
ROUND(x)	Menghasilkan pembulatan x, ROUND(6.7) = 7, ROUND(-6.7)=-7
ABS(x)	Menghasilkan nilai mutlak (absolut)
EXP(x)	Menghasilkan nilai $e^x$
MOD(x)	Menghasilkan sisa pembagian bilangan x dengan m. Berlaku untuk bilangan real
PI(x)	Menghasilkan nilai $\pi = 3.141593$
POW(x,y)	Menghasilkan nilai $x^y$

SQT(x)	Menghasilkan akar bilangan x, x harus positif
SIGN(x)	Menghasilkan nilai -1 kalau x negatif, 0 kalau x nol, 1 kalau x positif
TRUNCATE(x,d)	Pemotongan bilangan dengan d digit pecahan. Tidak ada pembulatan

Agar tampilan umur tidak mengandung pecahan, gunakan perintah berikut :

```
SELECT    nrp,    nama,    TRUNCATE (DATEDIFF (CURDATE () ,
    tgl_lahir)/365,0) as umur
FROM mahasiswa;
```

Perhatikan tampilan yang dihasilkan!

## 6. Tugas 7.2

Lakukan percobaan dengan menggunakan fungsi-fungsi yang lain!, dan laporkan ke asisten.

## 7. Fungsi Yang Lain

Fungsi	Keterangan
GREATEST(arg1, arg2,...)	Menghasilkan nilai terbesar dari argument-argumen / parameter
LEAST(arg1, arg2, ...)	Menghasilkan nilai terkecil dari argument-argumen / parameter
STRCMP(str1, str2)	Membandingkan 2 string, menghasilkan : <ul style="list-style-type: none"> <li>- 1 jika str1 = str2</li> <li>- 0 jika str1 != str2</li> <li>- Null jika ada argument bernilai null</li> </ul>
USER()	Menghasilkan user yang sedang menggunakan MySQL
DATABASE()	Database yang sedang digunakan
VERSION()	Menampilkan Versi MySQL

**A. Tujuan Pembelajaran**

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami dan menggunakan join pada MySQL
2. Menerapkan perintah join untuk mendapatkan informasi dari tabel

**B. Praktikum****1. Pendahuluan**

JOIN merupakan perintah di MySQL/MariaDB untuk menggabungkan 2 tabel atau lebih berdasarkan kolom yang sama.

JOIN di MySQL dibagi menjadi 3 cara :

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN

Sebelum kita melanjutkan pembahasan tentang join kita akan membuat database dengan nama perpustakaan yang terdiri dari 2 table yaitu table mahasiswa dan transaksi.

Tabel mahasiswa untuk menyimpan data mahasiswa

Tabel transaksi untuk menyimpan data transaksi peminjaman buku

**2. Membuat Database, Tabel dan Mengisi data**

Lakukan perintah

```
CREATE DATABASE perpustakaan; dan  
USE perpustakaan;
```

Buat dua tabel

```
CREATE TABLE mahasiswa  
(  nim INT(10),  
    nama VARCHAR(100),  
    alamat VARCHAR(100),  
    PRIMARY KEY(nim)  
);
```

```
CREATE TABLE transaksi  
(  id_transaksi INT(10),  
    nim INT(10),  
    buku VARCHAR(100),  
    PRIMARY KEY(id_transaksi)  
);
```

Masukan data untuk tabel mahasiswa dan transaksi

```
INSERT INTO mahasiswa
```

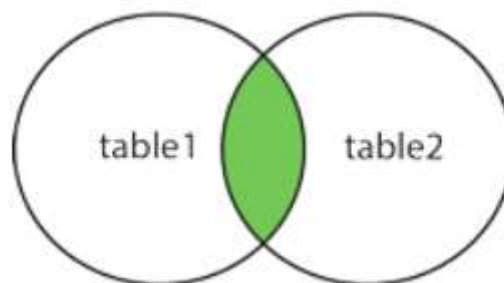
```
VALUES
(21400200,"faqih","bandung"),
(21400201,"ina","jakarta"),
(21400202,"anto","semarang"),
(21400203,"dani","padang"),
(21400204,"jaka","bandung"),
(21400205,"nara","bandung"),
(21400206,"senta","semarang");

INSERT INTO transaksi
VALUES
(1,21400200,"Buku Informatika"),
(2,21400202,"Buku Teknik Elektro"),
(3,21400203,"Buku Matematika"),
(4,21400206,"Buku Fisika"),
(5,21400207,"Buku Bahasa Indonesia"),
(6,21400210,"Buku Bahasa Daerah"),
(7,21400211,"Buku Kimia");
```

Tampilkan data mahasiswa dan transaksi dengan perintah select.

### 3. Inner Join

INNER JOIN membandingkan record di setiap table untuk dicek apakah nilai sama atau tidak.



Gambar 8.1 Inner Join

Jika nilai kedua table sama maka akan terbentuk table baru yang hanya menampilkan record yang sama dari kedua tabel.

Cara penulisannya :

```
SELECT *
FROM table1
INNER JOIN table2
ON table1.field = table2.field;
```

Contoh, Mencari data dari table mahasiswa dan transaksi berdasarkan kolom NIM

```
SELECT *
```

```

FROM mahasiswa
INNER JOIN transaksi
ON mahasiswa.nim = transaksi.nim;

```

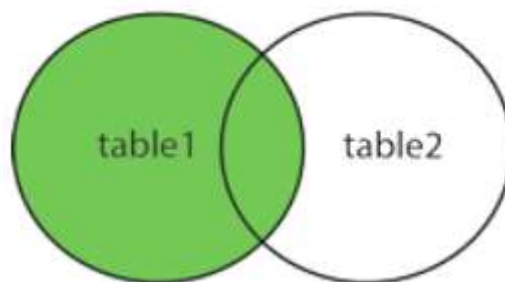
nim	nama	alamat	id_transaksi	nim	buku
21400200	faqih	bandung	1	21400200	Buku Informatika
21400202	anto	semarang	2	21400202	Buku Teknik Elektro
21400203	dani	padang	3	21400203	Buku Matematika
21400206	senta	semarang	4	21400206	Buku Fisika

Table mahasiswa mempunyai 7 record dan table transaksi mempunyai 7 record.

Jika menggabungkan kedua data menggunakan INNER JOIN berdasarkan kolom NIM maka hanya tampil 4 data mahasiswa yang meminjam buku di perpustakaan.

#### 4. Left Join

LEFT JOIN menghasilkan nilai berdasarkan table kiri (table1) dan nilai yang sama di table kanan (table2).



Gambar 8.2 Left Join

Jika table kanan tidak nilainya ada maka akan diisi nilai NULL

Cara Penulisan nya :

```

SELECT *
FROM table1
LEFT JOIN table2
ON table1.field = table2.field;

```

Contoh, Mencari data dari table mahasiswa dan transaksi berdasarkan kolom NIM

```

SELECT *
FROM mahasiswa
LEFT JOIN transaksi
ON mahasiswa.nim = transaksi.nim;

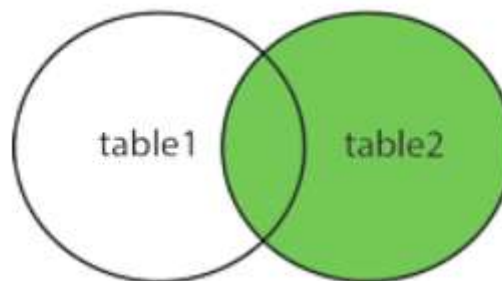
```

nim	nama	alamat	id_transaksi	nim	buku
21400200	faqih	bandung	1	21400200	Buku Informatika
21400202	anto	semarang	2	21400202	Buku Teknik Elektro
21400203	dani	padang	3	21400203	Buku Matematika
21400206	senta	semarang	4	21400206	Buku Fisika
21400201	ina	jakarta	NULL	NULL	NULL
21400204	jaka	bandung	NULL	NULL	NULL
21400205	nara	bandung	NULL	NULL	NULL

Table kiri (mahasiswa) akan menjadi table master dan mencari nilai yang sama di table transaksi. Apabila ada mahasiswa yang tidak meminjam buku maka diberi nilai NULL

## 5. Right Join

Konsep RIGHT JOIN hampir sama seperti LEFT JOIN hanya yang menjadi master adalah table kanan (table 2)



Gambar 8.3 Right Join

Jika table kiri tidak nilainya ada maka akan diisi nilai NULL

Cara penulisannya :

```
SELECT *
FROM table1
RIGHT JOIN table2
ON table1.field = table2.field;
```

Contoh, Mencari data dari table mahasiswa dan transaksi berdasarkan kolom NIM

```
SELECT *
FROM mahasiswa
RIGHT JOIN transaksi
ON mahasiswa.nim = transaksi.nim;
```

nim	nama	alamat	id_transaksi	nim	buku
21400200	faqih	bandung	1	21400200	Buku Informatika
21400202	anto	semarang	2	21400202	Buku Teknik Elektro
21400203	dani	padang	3	21400203	Buku Matematika
21400206	senta	semarang	4	21400206	Buku Fisika
NULL	NULL	NULL	5	21400207	Buku Bahasa Indonesia
NULL	NULL	NULL	6	21400210	Buku Bahasa Daerah
NULL	NULL	NULL	7	21400211	Buku Kimia

Terdapat 7 transaksi peminjaman buku di perpustakaan. Bagi transaksi yang NIM mahasiswa tidak ada di table mahasiswa akan diberi nilai NULL

#### A. Tujuan Pembelajaran

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami dan menggunakan store procedure
2. Menerapkan perintah query dalam store procedure

#### B. Praktikum

##### 1. Pendahuluan

Stored Procedure adalah sebuah fungsi berisi kode SQL yang dapat digunakan kembali.

Dalam Stored Procedure juga dapat dimasukkan parameter sehingga fungsi dapat digunakan lebih dinamis berdasarkan parameter tersebut.

Cara penulisan Stored Procedure

```
DELIMITER $$

CREATE PROCEDURE nama_procedure()
BEGIN
    kode sql
END$$

DELIMITER ;
```

Sedangkan untuk menjalankan Stored procedure adalah

```
CALL nama_procedure();
```

##### 2. Membuat Database dan Tabel Sederhana

Sebelum pembuatan store procedure, maka kita harus memiliki database dan tabelnya, database sebelumnya yang ada dapat digunakan, namun pada kesempatan ini dicontohkan secara sederhana pada sebuah tabel, dan kemudian Anda akan melanjutkan untuk melakukan explorasi dengan membuat store procedure sesuai kebutuhan dari tabel di praktikum sebelumnya.

Misalnya Anda membuat tabel mahasiswa dan memasukan datanya sebagai berikut:

```
CREATE TABLE mahasiswa
(
    nim INT(10),
    nama VARCHAR(100),
    alamat VARCHAR(100),
    PRIMARY KEY(nim)
);

INSERT INTO mahasiswa
```

```
VALUES
(21400200,"faqih","bandung"),
(21400201,"ina","jakarta"),
(21400202,"anto","semarang"),
(21400203,"dani","padang"),
(21400204,"jaka","bandung"),
(21400205,"nara","bandung"),
(21400206,"senta","semarang");
```

Lihat hasilnya dengan perintah select.

### 3. Membuat Store Procedure

Membuat Stored Procedure **selectMahasiswa()** untuk mendapatkan seluruh NIM dan NAMA mahasiswa.

```
DELIMITER $$

CREATE PROCEDURE selectMahasiswa()
BEGIN
    SELECT nim, nama FROM mahasiswa;
END$$

DELIMITER ;
```

Untuk memanggil store procedure selectMahasiswa():

```
CALL selectMahasiswa();
```

Contoh hasilnya :

nim	nama
21400200	faqih
21400201	ina
21400202	anto
21400203	dani
21400204	jaka
21400205	nara
21400206	senta

Jadi, setiap ingin menampilkan NIM dan NAMA mahasiswa kita tidak perlu membuat kode SQL seperti biasanya. Cukup simpan kode SQL di Stored Procedure dan bisa kita panggil dan gunakan berulang – ulang.

### 4. Store Procedure dengan Parameter

Kita juga memasukkan parameter di Stored Procedure agar menjadi lebih dinamis

Contoh, kita ingin membuat kode SQL untuk mencari data mahasiswa berdasarkan alamat.



```

DELIMITER $$

CREATE PROCEDURE alamatMahasiswa
(
    alamatMhs VARCHAR(100)
)
BEGIN
    SELECT *
    FROM mahasiswa
    WHERE alamat = alamatMhs;

END$$

DELIMITER ;

```

Di dalam nama Stored Procedure terdapat parameter alamatMhs dengan tipe data varchar

Saat masuk ke kode SQL yaitu mencari mahasiswa dengan alamat yang sama dengan parameter yang diinputkan.

Cara memanggilnya adalah

```
CALL alamatMahasiswa();
```

Hasilnya:

nim	nama	alamat
21400200	faqih	bandung
21400204	jaka	bandung
21400205	nara	bandung

Dengan Stored Procedure alamatMahasiswa() kita mencari data mahasiswa yang berasal dari “bandung”.

## 5. Data Manipulasi dengan Store Procedure

Penggunaan store procedure dapat dilakukan dengan menggabungkan data query language atau data manipulation language. Sebagai contoh kita ingin memasukan data mahasiswa dengan store procedure.

```

DELIMITER $$

CREATE PROCEDURE insertMahasiswa
(
    nimMhs INT(10),
    namaMhs VARCHAR(100),
    alamatMhs VARCHAR(100)
)
BEGIN
    INSERT INTO mahasiswa

```

```
VALUES (nimMhs, namaMhs, alamatMhs);

END$$

DELIMITER ;
```

Jika kita ingin memasukkan record baru di table mahasiswa maka akan ada 3 parameter saat memanggil Stored Procedure insertMahasiswa().

Contoh perintahnya :

```
CALL insertMahasiswa(21400207,"joni","jakarta");
```

Kemudian lakukan query untuk pengecekan nama di tabel mahasiswa.

```
SELECT * FROM mahasiswa WHERE namaMhs = "joni";
```

Selanjutnya jika ingin memasukkan data mahasiswa ke table mahasiswa tidak perlu membuat kode INSERT berkali-kali. Kita bisa gunakan Stored Procedure insertMahasiswa() untuk menggantikan proses INSERT yang biasanya kita gunakan.

Jadi Stored Procedure sangat penting dan akan memudahkan dalam menggunakan kode yang ingin dieksekusi secara berulang – ulang.

## 6. Tugas

- Buatlah beberapa store procedure yang menurut Anda sangat penting sehingga tidak perlu proses yang berulang dan sampaikan laporan hasil store prosedur Anda ke asisten. (hint : dapat menggunakan beberapa perintah dari praktikum modul sebelumnya)

**A. Tujuan Pembelajaran**

Setelah praktikum ini, praktikan diharapkan dapat :

1. Memahami dan menggunakan view pada MySQL

**B. Praktikum****1. Pendahuluan**

VIEW adalah perintah untuk membuat table virtual yang menyimpan kode SQL. Dengan view kita bisa membuat kode SQL yang kompleks dikemas menjadi satu table sederhana.

View akan menyimpan kode SQL yang kompleks tadi menjadi single table virtual yang lebih mudah untuk digunakan.

**2. Membuat VIEW**

Cara penulisan view sebagai berikut :

```
CREATE VIEW <nama view>
AS
Kode SQL
```

Saat kita mengeksekusi CREATE VIEW maka akan terbentuk table virtual yang menyimpan kode SQL.

Contoh, Kita akan membuat kode SQL yang menghubungkan table mahasiswa dan table transaksi secara INNER JOIN dan menyimpannya ke view. (**Lihat di Modul 8 Join untuk membuat database, tabel dan mengisi data-nya**). Jika sudah ada silahkan gunakan database perpustakaan dari modul 8.

Kita join table mahasiswa dan table transaksi berdasarkan field NIM secara INNER JOIN dengan perintah berikut :

```
SELECT mahasiswa.nim, nama, alamat, buku
FROM mahasiswa
INNER JOIN transaksi
ON mahasiswa.nim = transaksi.nim;
```

Hasilnya :

nim	nama	alamat	buku
21400200	faqih	bandung	Buku Informatika
21400202	anto	semarang	Buku Teknik Elektro
21400203	dani	padang	Buku Matematika
21400206	senta	semarang	Buku Fisika

Dengan view kita bisa membuat table virtual yang menyimpan query join di atas.  
Perintah untuk membuat view :

```
CREATE VIEW transaksiMhs AS
SELECT mahasiswa.nim, nama, alamat, buku
FROM mahasiswa
INNER JOIN transaksi
ON mahasiswa.nim = transaksi.nim
```

Jadi kita telah membuat table virtual dengan view dengan nama transaksiMhs. Cara menggunakannya adalah seperti melakukan query table biasa.

```
SELECT *
FROM transaksiMhs
```

Contoh lain, misal ingin query nama mahasiswa yang meminjam buku matematika.

```
SELECT nama, buku FROM transaksiMhs WHERE buku="Buku
Matematika";
```

nama	buku
dani	Buku Matematika

### 3. Menghapus VIEW

Cara penulisan untuk menghapus view yang sudah tidak digunakan.

```
DROP VIEW <nama view>;
```

Contoh menghapus view transaksiMhs

```
DROP VIEW transaksiMhs
```

### 4. Tugas

- Buatlah view untuk menampilkan data dari tabel transaksi atau tabel lainnya dari database yang sudah dibuat sebelumnya.
- Buat kesimpulan dari praktikum ini.