

**MODUL PRAKTIKUM
BASIS DATA**



Disusun oleh :
Ultach Enri, S.Kom., M.Kom

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2017**

DAFTAR ISI

Pertemuan 1 : MySQL	3
Pertemuan 2 : Tipe Data dan Arsitektur Basis Data	10
Pertemuan 3 : Perintah DDL.....	15
Pertemuan 4 : Perintah DML	23
Pertemuan 5 : Perintah DCL	29
Pertemuan 6 : Bahasa Query Formal Prosedural.....	36
Pertemuan 7 : Bahasa Query Formal Non Prosedural	47
Pertemuan 8 : Bahasa Query Formal Komersial.....	55
Pertemuan 9 : Fungsi Agregat.....	60
Daftar Pustaka	66

MySQL

1.1. Pendahuluan

MySQL adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya sangat cepat, merupakan program *multi-user* serta menggunakan perintah dasar SQL (*Structured Query Language*). MySQL merupakan dua bentuk lisensi, yaitu FreeSoftware dan Shareware. MySQL yang biasa kita gunakan adalah MySQL FreeSoftware yang berada dibawah Lisensi GNU/GPL (General Public License). MySQL pertama kali dirintis oleh seorang programmer database bernama Michael Widenius. Selain database server, MySQL juga merupakan program yang dapat mengakses suatu database MySQL yang berposisi sebagai *Server*, yang berarti program kita berposisi sebagai Client. Jadi MySQL adalah sebuah database yang dapat digunakan sebagai Client maupun server. Database MySQL merupakan suatu perangkat lunak database yang berbentuk database relasional atau disebut RDBMS (*Relational Database Management System*) yang menggunakan suatu bahasa permintaan yang bernama SQL (*Structured Query Language*).

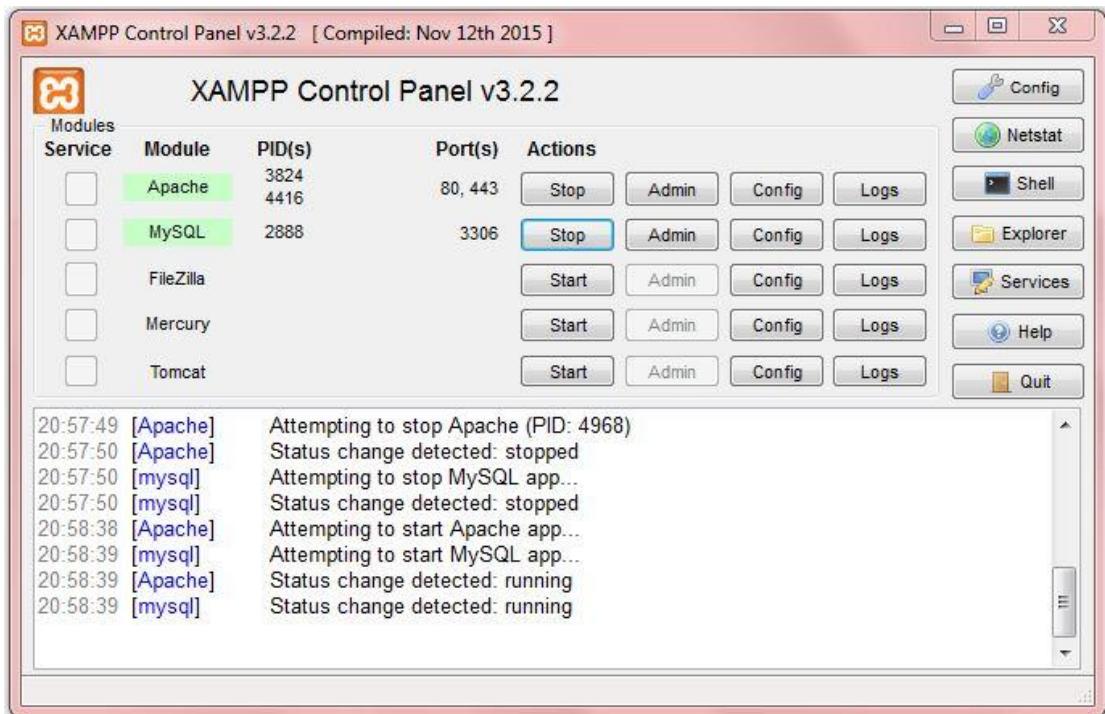
1.2. Kelebihan MySQL

Database MySQL memiliki beberapa kelebihan dibanding database lain, diantaranya :

- MySQL merupakan DBMS (*Database Management System*)
- MySQL sebagai RDBMS (*Relation Database Management System*)
- MySQL Merupakan sebuah *database server* yang free, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya
- MySQL dapat dijadikan sebagai database client
- MySQL mampu menerima *query* yang bertumpuk dalam satu permintaan atau *Multi-Threading*.
- MySQL merupakan database yang mampu menyimpan data berkapasitas sangat besar hingga berukuran Gigabyte.
- MySQL didukung oleh driver ODBC (*Open Database Connectivity*), artinya database MySQL dapat diakses menggunakan aplikasi apa saja termasuk berupa visual seperti visual Basic dan Delphi.
- MySQL adalah database menggunakan enkripsi password.
- MySQL merupakan Database Server yang multi user.
- MySQL mendukung field yang dijadikan sebagai kunci primer dan kunci unik.
- MySQL memiliki kecepatan dalam pembuatan tabel maupun updatetablel.

1.3. Mengaktifkan MySQL

Untuk dapat menggunakan MySQL terlebih dahulu aktifkan Server MySQL dengan mengaktifkan daemon MySQL. Program MySQL yang digunakan pada modul ini adalah XAMPP Control Panel v3.2.2.



Untuk masuk kedalam server MySQL, bukalah MS-DOS Prompt anda melalui Run kemudian ketik Command atau cmd. Maka anda dapat masuk ke dalam direktori MySQL melalui MS-DOS Promtp seperti dibawah ini:

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - mysql -u root". The window displays the MySQL monitor welcome message: "Welcome to the MariaDB monitor. Commands end with ; or \g. Your MariaDB connection id is 4 Server version: 10.1.8-MariaDB mariadb.org binary distribution Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement. MariaDB [(none)]>".

1.4. Masuk dan Keluar dari Server MySQL

MySQL adalah sebuah database server yang sangat aman. MySQL memiliki kemampuan memanajemen user dalam mengakses. Jadi, tidak sembarang user dapat mengakses sebuah database yang diciptakan MySQL. Maka sebelum anda memiliki User untuk mengakses MySQL anda juga dapat Mengakses database MySQL menggunakan User Root.

Berikut adalah perintah yang digunakan untuk mengkoneksikan kedalam Server Mysql :

Shell > mysql -u root

Keterangan : Tanda `-u` menerangkan bahwa kita akan masuk menggunakan User Name bernama `root`.

Untuk dapat keluar dari Server MySQL kita dapat mengetikkan Intruksi quit atau `\q`:

Mysql> quit

```
C:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 7
Server version: 10.1.8-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> quit
Bye
```

Mysql> \q

```
C:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.1.8-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> \q
Bye
```

1.5. Bantuan dalam MySQL

Database MySQL menyediakan beberapa fasilitas bantuan yang berguna untuk mendokumentasikan atau memanipulasikan server yaitu dengan cara mengetikan intruksi `\h` atau `\?`.

Mysql> \?

Opsional: Semua Query harus diakhiri dengan tanda titik koma (;). Tanda ini menunjukkan bahwa query telah berakhir dan siap dieksekusi.

Help (\h)	Digunakan untuk menampilkan file bantuan pada mysql.
? (\?)	Perintah ini sama dengan perintah Help.
Clear (\c)	Berguna untuk membersihkan atau menggagalkan semua perintah yang Telah berjalan dalam suatu prompt.
Connect (\r)	Untuk melakukan penyegaran koneksi ke dalam database yang ada pada Server Host.
Ego (\G)	Berguna untuk menampilkan data secara horizontal.
Go (\g)	Memberi perintah server untuk mengeksekusi.
tee (\T)	Mengatur tempat file yang akan didokumentasikan.

Note (\t)	Akhir dari (\T) yang berguna untuk mendokumentasikan semua query.
Print (\p)	Mencetak semua query yang telah kita perintahkan kelayar.
Prompt (\R)	Mengubah prompt standar sesuai keinginan.
Source (\.)	Berguna untuk mengeksekusi query dari luar yang berbentuk .sql
Use (\u)	Berguna untuk memasuki database yang akan digunakan maupun mengganti database yang akan digunakan.

1.6. Fungsi Pada Mysql

a. Fungsi String

- CONCAT (String 1, String 2, ..., String n)

Fungsi ini digunakan untuk menggabungkan dua atau lebih string (kolom).

```
MariaDB [(none)]> select concat ('Praktikum', ' ', 'Basis', ' ', 'Data');
+-----+
| concat ('Praktikum', ' ', 'Basis', ' ', 'Data') |
+-----+
| Praktikum Basis Data |
+-----+
1 row in set (0.00 sec)
```

- CONCAT_WS (separator, String 1, String 2, ..., String n)

Fungsi ini digunakan untuk menggabungkan dua atau lebih string (kolom) dengan separator (spasi).

```
MariaDB [(none)]> select concat_ws (' ', 'Praktikum', 'Basis', 'Data');
+-----+
| concat_ws (' ', 'Praktikum', 'Basis', 'Data') |
+-----+
| Praktikum Basis Data |
+-----+
1 row in set (0.00 sec)
```

- LENGTH (String)

Fungsi ini digunakan untuk menghitung panjang suatu string.

```
MariaDB [(none)]> select length ('Universitas Singaperbangsa Karawang');
+-----+
| length ('Universitas Singaperbangsa Karawang') |
+-----+
| 35 |
+-----+
1 row in set (0.00 sec)
```

- LEFT (String, Panjang)

Fungsi ini digunakan untuk memotong string dari sebelah kiri sebanyak panjang karakter.

```
MariaDB [(none)]> select left('Universitas Singaperbangsa Karawang', 5);
+-----+
| left('Universitas Singaperbangsa Karawang', 5) |
+-----+
| Unive |
+-----+
1 row in set (0.00 sec)
```

- RIGHT (String, Panjang)

Fungsi ini digunakan untuk memotong string dari sebelah kanan sebanyak panjang karakter.

```
MariaDB [none]> select right('Universitas Singaperbangsa Karawang', 5);
+-----+
| right('Universitas Singaperbangsa Karawang', 5) |
+-----+
| awang |
+-----+
1 row in set (0.00 sec)
```

- REPLACE (String, huruf awal, huruf pengganti)

Fungsi ini digunakan untuk mengganti suatu string dengan string yang lain.

```
MariaDB [none]> select replace ('www.unsika.ac.id','w','x');
+-----+
| replace ('www.unsika.ac.id','w','x') |
+-----+
| xxx.unsika.ac.id |
+-----+
1 row in set (0.00 sec)
```

- REPEAT (String, Jumlah)

Fungsi ini digunakan untuk menduplikasi suatu string sebanyak jumlah yang ditentukan.

```
MariaDB [none]> select repeat ('Unsika',5);
+-----+
| repeat ('Unsika',5) |
+-----+
| UnsikaUnsikaUnsikaUnsikaUnsika |
+-----+
1 row in set (0.00 sec)
```

b. Fungsi Waktu

- NOW()

Fungsi ini digunakan untuk mendapatkan waktu dan tanggal sekarang dari sistem.

```
MariaDB [none]> select now();
+-----+
| now() |
+-----+
| 2016-01-29 21:47:55 |
+-----+
1 row in set (0.06 sec)
```

- YEAR (now())

Fungsi ini digunakan untuk mendapatkan tahun sekarang dari sistem.

```
MariaDB [none]> select year (now());
+-----+
| year (now()) |
+-----+
| 2016 |
+-----+
1 row in set (0.00 sec)
```

1.7. Administrasi MySQL

MySQL Selaku database server yang mampu berjalan pada jaringan, tentu saja MySQL harus memiliki kemampuan khusus yang berguna untuk melakukan manajemen user atau mendukung system database yang bersifat client/server.

a. Membuat User baru

Untuk dapat menciptakan user baru pada database mysql yang terdapat pada tabel user. Dapat dilakukan dengan menggunakan pernyataan SQL bernama CREATE.

Sintax seperti berikut :

```
CREATE USER nama_user;
```

- nama_user adalah nama dari user yang akan dibuat, maksimal 16 karakter.

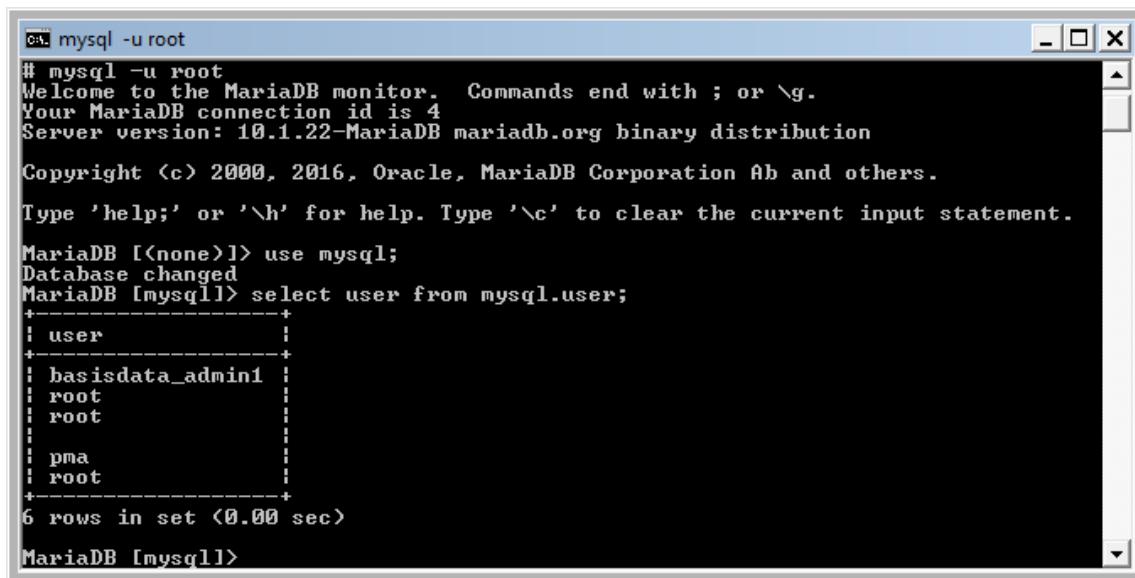
```
MariaDB [⟨none⟩] > create user basisdata_admin1;
Query OK, 0 rows affected (0.10 sec)
```

b. Melihat Daftar User

Pengguna mysql dapat melihat user lainnya yang telah diciptakan sebelumnya. Hal ini digunakan untuk melihat siapa saja pengguna dari database tersebut.

Sintax seperti berikut :

```
SELECT * FROM mysql.user;
```



user	Host
basisdata_admin1	%
root	localhost
root	127.0.0.1
pma	localhost
root	127.0.0.1

6 rows in set (0.00 sec)

```
MariaDB [mysql]>
```

c. Menghapus User

Sintax seperti berikut :

```
DROP user nama_user;
```

- nama_user adalah nama dari user yang akan dihapus.

Tugas

1. Buatlah teks sebagai berikut:
“MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more. Oracle drives MySQL innovation, delivering new capabilities to power next generation web, cloud, mobile and embedded applications.”
2. Hitung panjang karakter teks tersebut.
3. Potong karakter bagian kiri sebanyak 113.
4. Potong karakter bagian kanan sebanyak 155.
5. Ganti karakter '.' (titik) menjadi '\\\' (slash 2x).
6. Buatlah user sebagai berikut:
 - dosen@localhost
 - mahasiswa@localhost
 - pustakawan@localhost
 - admin@localhost
 - security@localhost
7. Tampilkan semua user.
8. Hapus user security@localhost

Tipe Data dan Arsitektur Basis Data

2.1 Tipe Data Pada Mysql

Tipe data adalah suatu bentuk pemodelan data yang dideklarasikan pada saat melakukan pembuatan tabel. Tipe data ini akan mempengaruhi setiap data yang akan dimasukkan ke dalam sebuah tabel. Data yang akan dimasukkan harus sesuai dengan tipe data yang dideklarasikan.

Berbagai tipe data pada MySQL dapat dilihat pada tabel berikut :

Type Data	Keterangan
TINYINT	Ukuran 1 byte. Bilangan bulat terkecil, dengan jangkauan untuk bilangan bertanda: -128 sampai dengan 127 dan untuk yang tidak bertanda : 0 s/d 255. Bilangan tak bertandai dengan kata UNSIGNED
SMALLINT	Ukuran 2 Byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -32768 s/d 32767 dan untuk yang tidak bertanda : 0 s/d 65535
MEDIUMINT	Ukuran 3 byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -8388608 s/ d 8388607 dan untuk yang tidak bertanda : 0 s/d 16777215
INT	Ukuran 4 byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -2147483648 s/d 2147483647 dan untuk yang tidak bertanda : 0 s/d 4294967295
INTEGER	Ukuran 4 byte. Sinonim dari int
BIGINT	Ukuran 8 byte. Bilangan bulat terbesar dengan jangkauan untuk bilangan bertanda : -9223372036854775808 s/d 9223372036854775807 dan untuk yang tidak bertanda : 0 s/d 1844674473709551615
FLOAT	Ukuran 4 byte. Bilangan pecahan
DOUBLE	Ukuran 8 byte. Bilangan pecahan
DOUBLEPRECISION	Ukuran 8 byte. Bilangan pecahan
REAL	Ukuran 8 byte. Sinonim dari DOUBLE
DECIMAL (M,D)	Ukuran M byte. Bilangan pecahan, misalnya DECIMAL(5,2) dapat digunakan untuk menyimpan bilangan -99,99 s/d 99,99
NUMERIC (M,D)	Ukuran M byte. Sinonim dari DECIMAL, misalnya NUMERIC(5,2) dapat digunakan untuk menyimpan bilangan -99,99 s/d 99,99

Type Data untuk Bilangan (Number)

Type Data	Keterangan
DATETIME	Ukuran 8 byte. Kombinasi tanggal dan jam, dengan jangkauan dari '1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'
DATE	Ukuran 3 Byte. Tanggal dengan jangkauan dari '1000-01-01' s/d '9999-12-31'
TIMESTAMP	Ukuran 4 byte. Kombinasi tanggal dan jam, dengan jangkauan dari '1970-01-01 00:00:00' s/d '2037'
TIME	Ukuran 3 byte. Waktu dengan jangkauan dari '839:59:59' s/d '838:59:59'
YEAR	Ukuran 1 byte. Data tahun antara 1901 s/d 2155

Type Data untuk Tanggal dan Jam

Type Data	Keterangan
CHAR	Mampu menangani data hingga 255 karakter. Tipe data CHAR mengharuskan untuk memasukkan data yang telah ditentukan oleh kita.
VARCHAR	Mampu menangani data hingga 255 karakter. Tipe data VARCHAR tidak mengharuskan untuk memasukkan data yang telah ditentukan oleh kita.
TINYBLOB, TINYTEXT	Ukuran 255 byte. Mampu menangani data sampai 2^{8-1} data.
BLOB, TEXT	Ukuran 65535 byte. Type string yang mampu menangani data hingga 2^{16-1} (16M-1) data.
MEDIUMBLOB, MEDIUMTEXT	Ukuran 16777215 byte. Mampu menyimpan data hingga 2^{24-1} (16M-1) data.
LONGBLOB, LONGTEXT	Ukuran 4294967295 byte. Mampu menyimpan data hingga berukuran GIGA BYTE. Tipe data ini memiliki batas penyimpanan hingga 2^{32-1} (4G-1) data.
ENUM('nilai1','nilai2',...,'nilaiN')	Ukuran 1 atau 2 byte. Tergantung jumlah nilai enumerasinya (maksimum 65535 nilai)
SET('nilai1','nilai2',...,'nilaiN')	1,2,3,4 atau 8 byte, tergantung jumlah anggota himpunan (maksimum 64 anggota)

Type Data untuk Karakter dan Lain-lain

Perbedaan antara **CHAR dan **VARCHAR** adalah dari cara MySQL mengalokasikan ukuran penyimpanan data yang diinput kedalam kolom tersebut. Contohnya, jika kita mendefinisikan sebuah tabel dengan kolom bertipe CHAR(5), walaupun huruf atau karakter yang kita inputkan hanya 1 karakter, MySQL tetap menyimpan kolom tersebut untuk 5 karakter. Namun jika kita definiskan sebagai VARCHAR(5), dan kita menginput data dengan jumlah karakter 2, maka ukuran penyimpanan hanya akan menggunakan 2 karakter, sehingga VARCHAR lebih fleksibel dan efisien.

2.2 Arsitektur Basis Data

Structure Query Language

SQL (Structured Query Language) adalah sebuah bahasa permintaan database yang terstruktur. Bahasa SQL ini dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam database maupun merelasikan antar database.

SQL dibagi menjadi tiga bentuk Query, yaitu :

1. DDL (Data Definition Language)

DDL adalah sebuah metode Query SQL yang berguna untuk mendefinisikan data pada sebuah Database.

Query yang dimiliki DDL adalah :

- CREATE : Digunakan untuk membuat Database dan Tabel.
- Drop : Digunakan untuk menghapus Tabel dan Database.
- Alter : Digunakan untuk melakukan perubahan struktur tabel yang telah dibuat, baik menambah Field (Add), mengganti nama Field (Change) ataupun mengganti nama tabel (Rename), dan menghapus Field (Drop).

2. DML (Data Manipulation Language)

DML adalah sebuah metode Query yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari Query DML ini untuk melakukan pemanipulasi database yang telah dibuat. Query yang dimiliki DML adalah :

- INSERT : Digunakan untuk memasukkan data pada tabel Database.
- UPDATE : Digunakan untuk pengubahan terhadap data yang ada pada tabel Database.
- DELETE : Digunakan untuk penghapusan data pada tabel Database.

3. DCL (Data Control Language)

DCL adalah sebuah metode Query SQL yang digunakan untuk memberikan hak otorisasi mengakses Database, mengalokasikan space, pendefinisian space, dan pengauditan penggunaan database.

Query yang dimiliki DCL adalah :

- GRANT : Untuk mengizinkan User mengakses Tabel dalam Database.
- REVOKE : Untuk membatalkan izin hak user, yang ditetapkan oleh perintah GRANT.
- COMMIT : Mempertahankan penyimpanan Database.
- ROLLBACK : Membatalkan penyimpanan Database.

Latihan

1. Membuat Database Dan Tabel Sederhana

Database adalah sebuah media utama yang harus dibuat dalam membangun sebuah basis data agar nantinya dapat kita letakkan beberapa tabel dengan field-fieldnya.

Sintaks sebagai berikut :

CREATE DATABASE 'nama_database';

Buatlah database dengan nama rumahsakit

```

MariaDB [(none)]> create database rumahsakit;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| kuliah |
| mahasiswa |
| mysql |
| performance_schema |
| phi_minimart |
| phpmyadmin |
| puskesmas |
| rs |
| rumah_sakit |
| rumahsakit |
| test |
+-----+
12 rows in set (0.00 sec)

```

2. Menghapus Database

Untuk menghapus Database yang telah dibuat dapat menggunakan query SQL berikut :

DROP DATABASE nama_database;

Hapus database rumahsakit

```

MariaDB [(none)]> drop database rumahsakit;
Query OK, 11 rows affected (0.13 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phi_minimart |
| phpmyadmin |
| rumah_sakit |
| test |
+-----+
7 rows in set (0.00 sec)

```

3. Melihat struktur tabel

- Sebelum melihat struktur tabel, buat kembali database dengan nama **rumah_sakit**.
- selanjutnya masukkan query **show databases**; untuk melihat apakah database rumah_sakit sudah tersedia.
- Aktifkan database dengan menggunakan sintaks:
use 'nama_database';
contoh: **use rumah_sakit;**

```

MariaDB [(none)]> use rumah_sakit;
Database changed
MariaDB [rumah_sakit]>

```

- Selanjutnya adalah **membuat tabel pasien** dengan field sebagai berikut:

```

MariaDB [rumah_sakit]> create table pasien (
    -> kd_pasien int<11>primary key,
    -> nama_pasien varchar<30>;
Query OK, 0 rows affected (0.52 sec)

```

- Setelah tabel dibuat, anda dapat melihat tipe data dan panjang recordset dengan cara menampilkan struktur tabel.

Sintaks sebagai berikut:

DESC nama_tabel;

Atau

```

DESCRIBE nama_tabel;
MariaDB [rumah_sakit]> desc pasien;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| kd_pasien | int(11) | NO | PRI | NULL |       |
| nama_pasien | varchar(30) | YES |     | NULL |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

4. Menghapus tabel

Untuk menghapus Tabel yang telah dibuat dapat menggunakan query SQL berikut :

```
DROP TABLE nama_tabel;
```

```
MariaDB [rumah_sakit]> drop table pasien;
Query OK, 0 rows affected (0.18 sec)
```

Tugas Pertemuan 2 (Take home)

1. Rancang database kuliah terdiri dari 3 tabel:

1.1 Tabel matkul

- (a) Kode_matkul char (5) primary key
- (b) nama_matkul varchar (30) not null
- (c) semester int (1) not null
- (d) sks int (1) not null
- (e) prodi varchar (20) not null

1.2 Tabel dosen

- (a) kd_dosen char (10) primary key
- (b) nama_dosen varchar (30) not null
- (c) tgl_lahir date not null
- (d) jenis_kelamin varchar (2) not null
- (e) alamat varchar (100)not null
- (f) no_telp varchar (15)not null

1.3 Tabel mahasiswa

- (a) nim char (10) primary key
- (b) nama varchar (30) not null
- (c) tanggal_lahir date not null
- (d) jenis_kelamin char (2) not null
- (e) jurusan varchar (20) not null
- (f) alamat varchar (100) not null

- 2) Tampilkan struktur tabel:

- (a) mahasiswa
- (b) matkul
- (c) dosen

- 3) Hapus tabel dosen

PERINTAH DDL

1. Mengganti Nama Tabel

MySQL menyediakan query **ALTER...RENAME TO** dengan format query:

```
ALTER TABLE nama_tabel_lama RENAME TO nama_tabel_baru;
```

Selain query **ALTER...RENAME TO**, terdapat juga perintah **RENAME** untuk merubah nama tabel,format querynya:

```
RENAME TABLE nama_tabel_lama TO nama_tabel_baru;
```

```
MariaDB [rumah_sakit]> show tables;
+-----+
| Tables_in_rumah_sakit |
+-----+
| dokter
| obat
| pasien
| periksa
| resep
+-----+
5 rows in set (0.00 sec)

MariaDB [rumah_sakit]> alter table periksa rename to periksa_pasien;
Query OK, 0 rows affected (0.28 sec)

MariaDB [rumah_sakit]> show tables;
+-----+
| Tables_in_rumah_sakit |
+-----+
| dokter
| obat
| pasien
| periksa_pasien
| resep
+-----+
5 rows in set (0.00 sec)
```

2. Menambah Kolom Pada Tabel

```
ALTER TABLE nama_tabel ADD nama_kolom_baru tipe_data;
```

a) Tabel pasien

1. Tambahkan kolom tempat lahir setelah kolom jenis kelamin

```
MariaDB [rumah_sakit]> alter table pasien
    -> add tempat_lahir varchar(20)
    -> after jenis_kelamin;
Query OK, 0 rows affected (0.66 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc pasien;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| kd_pasien | int(11) | NO   | PRI  | NULL    |       |
| nama      | varchar(30) | YES  | MUL  | NULL    |       |
| jenis_kelamin | char(10) | YES  |      | NULL    |       |
| tempat_lahir | varchar(20) | YES  |      | NULL    |       |
| tgl_lahir  | date    | YES  |      | NULL    |       |
| alamat     | varchar(50) | YES  |      | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

2. Tambahkan kolom usia setelah kolom alamat

```
MariaDB [rumah_sakit]> alter table pasien
    -> add usia varchar(2)
    -> after alamat;
Query OK, 0 rows affected (0.59 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc pasien;
```

Field	Type	Null	Key	Default	Extra
kd_pasien	int(11)	NO	PRI	NULL	
nama	varchar(30)	YES	MUL	NULL	
jenis_kelamin	char(10)	YES		NULL	
tempat_lahir	varchar(20)	YES		NULL	
tgl_lahir	date	YES		NULL	
alamat	varchar(50)	YES		NULL	
usia	varchar(2)	YES		NULL	

7 rows in set (0.00 sec)

b) Tabel dokter

1. Tambahkan kolom no_telp setelah kolom email

```
MariaDB [rumah_sakit]> alter table dokter
    -> add no_telp varchar(15) not null
    -> after email;
Query OK, 0 rows affected (0.47 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc dokter;
```

Field	Type	Null	Key	Default	Extra
kd_dokter	int(11)	NO	PRI	NULL	
nama_dokter	varchar(30)	NO		NULL	
spesialis	varchar(30)	NO		NULL	
jenis_kelamin	char(10)	YES		NULL	
alamat	varchar(50)	NO		NULL	
email	varchar(30)	NO		NULL	
no_telp	varchar(15)	NO		NULL	

7 rows in set (0.01 sec)

2. Tambahkan kolom status setelah kolom spesialis

```
MariaDB [rumah_sakit]> alter table dokter
    -> add status varchar(10) not null
    -> after spesialis;
Query OK, 0 rows affected (0.52 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc dokter;
```

Field	Type	Null	Key	Default	Extra
kd_dokter	int(11)	NO	PRI	NULL	
nama_dokter	varchar(30)	NO		NULL	
spesialis	varchar(30)	NO		NULL	
status	varchar(10)	NO		NULL	
jenis_kelamin	char(10)	YES		NULL	
alamat	varchar(50)	NO		NULL	
email	varchar(30)	NO		NULL	
no_telp	varchar(15)	NO		NULL	

8 rows in set (0.01 sec)

c) Tabel periksa_pasien

Tambahkan kolom biaya setelah kolom tindakan

```
MariaDB [rumah_sakit]> alter table periksa_pasien  
    -> add biaya int <15> not null  
        -> after tindakan;  
Query OK, 0 rows affected (0.71 sec)  
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc periksa_pasien;
```

Field	Type	Null	Key	Default	Extra
kd_periksa	varchar(11)	NO	PRI	NULL	
kd_dokter	int(11)	NO	MUL	NULL	
kd_pasien	int(11)	NO	MUL	NULL	
diagnosa	varchar(50)	NO		NULL	
tingdakan	varchar(50)	NO		NULL	
biaya	int(15)	NO		NULL	

6 rows in set (0.00 sec)

d) Tabel obat

Tambahkan kolom harga setelah kolom tgl_expired

```
MariaDB [rumah_sakit]> alter table obat  
    -> add harga int <15> not null  
        -> after tgl_expired;  
Query OK, 0 rows affected (0.45 sec)  
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc obat;
```

Field	Type	Null	Key
Default	Extra		
kd_obat	varchar(5)	NO	PRI
NULL			
nama	varchar(20)	NO	
NULL			
jenis	enum('tablet','serbut','pil','kapsul','larutan')	NO	
NULL			
stok	int(4)	NO	
NULL			
tgl_expired	date	NO	
NULL			
harga	int<15>	NO	
NULL			

6 rows in set (0.01 sec)

e) Tabel resep

Tambahkan kolom jumlah_obat setelah kolom aturan_pakai

```
MariaDB [rumah_sakit]> desc resep;
```

Field	Type	Null	Key	Default	Extra
no_resep	varchar(5)	YES		NULL	
tgl_resep	date	NO		NULL	
aturan_pakai	varchar(5)	NO		NULL	
jumlah_obat	int(3)	NO		NULL	

4 rows in set (0.05 sec)

3. Merubah Nama Kolom Tabel

Untuk merubah nama kolom pada tabel yang sudah ada, dapat menggunakan perintah ALTER...CHANGE, dengan format query sebagai berikut:

```
ALTER TABLE nama_tabel CHANGE nama_kolom nama_kolom_baru tipe_data;
```

a) Tabel pasien

Ubah nama kolom jenis_kelamin menjadi jk

```
MariaDB [rumah_sakit]> alter table pasien  
    -> change jenis_kelamin jk char(10);  
Query OK, 0 rows affected (0.08 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [rumah_sakit]> desc pasien;
```

Field	Type	Null	Key	Default	Extra
kd_pasien	int(11)	NO	PRI	NULL	
nama	varchar(30)	YES	MUL	NULL	
jk	char(10)	YES		NULL	
tempat_lahir	varchar(20)	YES		NULL	
tgl_lahir	date	YES		NULL	
alamat	varchar(50)	YES		NULL	
usia	varchar(2)	YES		NULL	

7 rows in set (0.00 sec)

Ubah nama kolom tempat_lahir menjadi tempat_tinggal

```
MariaDB [rumah_sakit]> alter table pasien change tempat_lahir tempat_tinggal var  
char(20);  
Query OK, 0 rows affected (0.05 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [rumah_sakit]> desc pasien;
```

Field	Type	Null	Key	Default	Extra
kd_pasien	int(11)	NO	PRI	NULL	
nama	varchar(30)	YES	MUL	NULL	
jk	enum('pria','wanita')	YES		NULL	
tempat_tinggal	varchar(20)	YES		NULL	
tgl_lahir	date	YES		NULL	
alamat	varchar(50)	YES		NULL	
usia	varchar(2)	YES		NULL	

7 rows in set (0.02 sec)

b) Tabel dokter

Ubah nama kolom jenis_kelamin menjadi jk dan tipe data menjadi enum ('pria,'wanita')

```
MariaDB [rumah_sakit]> alter table dokter  
    -> change jenis_kelamin jk enum ('pria','wanita');  
Query OK, 0 rows affected (0.08 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [rumah_sakit]> desc dokter;
```

Field	Type	Null	Key	Default	Extra
kd_dokter	int(11)	NO	PRI	NULL	
nama_dokter	varchar(30)	NO		NULL	
spesialis	varchar(30)	NO		NULL	
status	varchar(10)	NO		NULL	
jk	enum('pria','wanita')	YES		NULL	
alamat	varchar(50)	NO		NULL	
email	varchar(30)	NO		NULL	
no_telp	varchar(15)	NO		NULL	

8 rows in set (0.01 sec)

c) Tabel periksa_pasien

Ubah nama kolom biaya menjadi biaya_tindakan

```
MariaDB [rumah_sakit]> alter table periksa_pasien  
-> change biaya biaya_tindakan int <15> not null;  
Query OK, 0 rows affected (0.09 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [rumah_sakit]> desc periksa_pasien;
```

Field	Type	Null	Key	Default	Extra
kd_periksa	varchar(11)	NO	PRI	NULL	
kd_dokter	int<11>	NO	MUL	NULL	
kd_pasien	int<11>	NO	MUL	NULL	
diagnosa	varchar<50>	NO		NULL	
tindakan	varchar<50>	NO		NULL	
biaya_tindakan	int<15>	NO		NULL	

6 rows in set (0.00 sec)

d) Tabel obat

Ubah nama menjadi nama_obat

```
MariaDB [rumah_sakit]> alter table obat  
-> change nama nama_obat varchar <20> not null;  
Query OK, 0 rows affected (0.03 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [rumah_sakit]> desc obat;
```

Field	Type	Null	Key
Default	Extra		
kd_obat	varchar<5>	NO	PRI
NULL			
nama_obat	varchar<20>	NO	
NULL			
jenis	enum('tablet','serbut','pil','kapsul','larutan')	NO	
NULL			
stok	int<4>	NO	
NULL			
tgl_expired	date	NO	
NULL			
harga	int<15>	NO	
NULL			

6 rows in set (0.00 sec)

4. Menghapus Kolom Pada Tabel

Kebalikan dari menambahkan kolom baru, query ALTER..DROP dapat digunakan untuk menghapus sebuah kolom dari tabel MySQL.

Sintaks sebagai berikut:

```
ALTER TABLE nama_tabel DROP nama_kolom;
```

```
Hapus kolom status pada tabel dokter
```

```
MariaDB [rumah_sakit]> alter table dokter drop status;
Query OK, 0 rows affected (0.64 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc dokter;
```

Field	Type	Null	Key	Default	Extra
kd_dokter	int(11)	NO	PRI	NULL	
nama_dokter	varchar(30)	NO		NULL	
spesialis	varchar(30)	NO		NULL	
jk	enum('pria','wanita')	YES		NULL	
alamat	varchar(50)	NO		NULL	
email	varchar(30)	NO		NULL	
no_telp	varchar(15)	NO		NULL	

```
7 rows in set (0.00 sec)
```

5. Merubah Ukuran/Tipe Pada Kolom

Berikut merupakan query ALTER....MODIFY dapat digunakan untuk mengubah sebuah tipe data pada sebuah kolom.

Sintaks sebagai berikut:

```
ALTER TABLE nama_tabel MODIFY nama_kolom tipe_data_baru;
```

Ubah tipe data jk pada tabel pasien dengan enum ('pria','wanita')

```
MariaDB [rumah_sakit]> alter table pasien modify jk enum ('pria','wanita');
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [rumah_sakit]> desc pasien;
```

Field	Type	Null	Key	Default	Extra
kd_pasien	int(11)	NO	PRI	NULL	
nama	varchar(30)	YES	MUL	NULL	
jk	enum('pria','wanita')	YES		NULL	
tempat_tinggal	varchar(20)	YES		NULL	
tgl_lahir	date	YES		NULL	
alamat	varchar(50)	YES		NULL	
usia	varchar(2)	YES		NULL	

```
7 rows in set (0.02 sec)
```

6. Menambah Nilai Default

Berikut merupakan bagian dari query ALTER....MODIFY dengan mengatur nilai default pada sebuah kolom.

Sintaks sebagai berikut:

```
ALTER TABLE nama_tabel MODIFY nama_kolom tipe_data defaul 'value';
```

Menambah nilai defaul stok = 0 di tabel obat

```

MariaDB [rumah_sakit]> alter table obat
    -> modify stok int <4> not null default '0';
Query OK, 0 rows affected <0.00 sec>
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [rumah_sakit]> desc obat;
+-----+-----+-----+-----+
| Field | Type  | Null | Key |
|-----+-----+-----+-----+
| kd_obat | varchar(5) | NO  | PRI |
| NULL |          |      |      |
| nama_obat | varchar(20) | NO  |      |
| NULL |          |      |      |
| jenis | enum('tablet','serbut','pil','kapsul','larutan') | NO  |      |
| NULL |          |      |      |
| stok | int<4> | NO  |      |
| 0 |          |      |      |
| tgl_expired | date | NO  |      |
| NULL |          |      |      |
| harga | int<15> | NO  |      |
| NULL |          |      |      |
+-----+-----+-----+-----+
6 rows in set <0.00 sec>

```

7. Menambah Primary Dan Foreign Key

a) Menambahkan primary key

Berikut merupakan cara apabila kita lupa menambahkan primary key pada kolom dalam sebuah tabel.

Sintaks sebagai berikut:

```
ALTER TABLE nama_tabel ADD PRIMARY KEY(nama_kolom);
```

Menambahkan primary key di tabel resep pada kolom no_resep

```

MariaDB [rumah_sakit]> alter table resep add primary key (no_resep);
Query OK, 0 rows affected <0.67 sec>
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [rumah_sakit]> desc resep;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
|-----+-----+-----+-----+-----+-----+
| no_resep | varchar(5) | NO  | PRI | NULL   |      |
| tgl_resep | date   | NO  |      | NULL   |      |
| aturan_pakai | varchar(5) | NO  |      | NULL   |      |
| jumlah_obat | int(3) | NO  |      | NULL   |      |
+-----+-----+-----+-----+-----+-----+
4 rows in set <0.00 sec>

```

b) Menambahkan foreign key

Berikut merupakan cara apabila kita ingin menambahkan foreign key pada kolom dalam sebuah tabel.

Sintaks sebagai berikut:

```
ALTER TABLE nama_tabel ADD FOREIGN KEY (nama_kolom) REFERENCES nama_tabel_referensi (PK nama_kolom_referensi) ON DELETE CASCADE ON UPDATE RESTRICT;
```

Sebelumnya tambahkan kolom kd_periksa (pada tabel periksa pasien) dan kolom kd_obat (pada tabel obat) ke tabel resep

```

MariaDB [rumah_sakit]> alter table resep
    -> add kd_periksa varchar (11) not null
    -> after no_resep;
Query OK, 0 rows affected <0.48 sec>
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [rumah_sakit]> alter table resep
    -> add kd_obat varchar (5) not null
    -> after kd_periksa;
Query OK, 0 rows affected <0.71 sec>
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [rumah_sakit]> desc resep;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| no_resep | varchar(5) | NO   | PRI   | NULL    |       |
| kd_periksa | varchar(11) | NO   |       | NULL    |       |
| kd_obat | varchar(5) | NO   |       | NULL    |       |
| tgl_resep | date    | NO   |       | NULL    |       |
| aturan_pakai | varchar(5) | NO   |       | NULL    |       |
| jumlah_obat | int(3)  | NO   |       | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set <0.00 sec>

```

Tambahkan foreign key pada kd_periksa dan kd_obat di tabel resep

```

MariaDB [rumah_sakit]> alter table resep
    -> add foreign key (kd_periksa)
    -> references periksa_pasien (kd_periksa)
    -> on delete cascade
    -> on update cascade;
Query OK, 0 rows affected <0.86 sec>
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [rumah_sakit]> alter table resep
    -> add foreign key (kd_obat)
    -> references obat (kd_obat)
    -> on delete cascade
    -> on update cascade;
Query OK, 0 rows affected <1.34 sec>
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [rumah_sakit]> desc resep;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| no_resep | varchar(5) | NO   | PRI   | NULL    |       |
| kd_periksa | varchar(11) | NO   | MUL   | NULL    |       |
| kd_obat | varchar(5) | NO   | MUL   | NULL    |       |
| tgl_resep | date    | NO   |       | NULL    |       |
| aturan_pakai | varchar(5) | NO   |       | NULL    |       |
| jumlah_obat | int(3)  | NO   |       | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set <0.00 sec>

```

***CASCADE**, Baris-baris dalam tabel anak yang berisi nilai-nilai yang juga terdapat dalam kolom terkait dari tabel induk dihapus ketika baris-baris yang berkaitan dihapus dari tabel induk. Baris-baris dalam tabel anak yang berisi nilai-nilai yang juga terdapat dalam kolom terkait dari tabel induk diupdate ketika nilai-nilai yang berkaitan diupdate dalam tabel induk. **JIKA TERJADI PERUBAHAN PADA TABEL INDUK MAKA TABEL FOREIGN KEY MENGAKIBARKAN TERJADI PERUBAHAN.**

PERINTAH DML

Data Manipulation Language

DML berfungsi untuk memanipulasi data dalam database yang telah dibuat. Perintah-perintah yang digunakan diantaranya :

- INSERT : menyisipkan atau menambahkan data baru kedalam tabel
- SELECT : mengambil atau menampilkan data dari tabel.
- UPDATE: memperbarui data yang lama ke data yang baru.
- DELETE : menghapus data dalam tabel.

1. Perintah insert

Memasukkan satu record ke dalam tabel:

```
INSERT INTO nama_tabel VALUES ('nilai1', 'nilai 2', 'nilai 3');
```

Memasukkan lebih dari 1 record ke dalam tabel:

```
INSERT INTO nama_tabelVALUES
```

```
-> ('value11','value12','value13','...','value n'),  
-> ('value21','value22','value23','...','value n'),  
-> ('value n1','value n2','value n3','...','value nn');
```

a) Tabel pasien

```
MariaDB [rumah_sakit]> insert into pasien values  
-> ('10001','ardi','pria','jakarta','1992-05-02','jl. nangka no 10','23'),  
-> ('10002','firna','wanita','bekasi','1995-01-01','jl. pekan raya no 90','21'),  
-> ('10003','jeffrey','pria','bandung','1995-09-11','jl. raya cakung no 1','21'),  
-> ('10004','glenn','pria','jakarta','2000-12-12','jl. tanjungan no 13','15'),  
-> ('10005','asri','wanita','bekasi','1996-11-10','jl. merdeka no 89','20');  
  
Query OK, 5 rows affected (0.08 sec)  
Records: 5 Duplicates: 0 Warnings: 0  
  
MariaDB [rumah_sakit]> select * from pasien;  
+-----+-----+-----+-----+-----+  
| kd_pasien | nama | jk | tempat_lahir | tgl_lahir | alamat |  
| usia |  
+-----+-----+-----+-----+-----+  
| 10001 | ardi | pria | jakarta | 1992-05-02 | jl. nangka no 10 |  
| 23 |  
| 10002 | firna | wanita | bekasi | 1995-01-01 | jl. pekan raya no 9 |  
| 21 |  
| 10003 | jeffrey | pria | bandung | 1995-09-11 | jl. raya cakung no |  
| 21 |  
| 10004 | glenn | pria | jakarta | 2000-12-12 | jl. tanjungan no 13 |  
| 15 |  
| 10005 | asri | wanita | bekasi | 1996-11-10 | jl. merdeka no 89 |  
| 20 |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

b) Tabel dokter

```
MariaDB [rumah_sakit]> insert into dokter values
    -> ('50001','dr. Marini, Sp.A','anak','wanita','jl. merdeka no 11','marini@gmail.com','089090909090'),
    -> ('50002','dr. Eddy, Sp.P','paru','pria','jl. juang no 87','eddy@gmail.com','087979797979'),
    -> ('50003','dr. Santoso, Sp.P','paru','pria','jl. kalimalang no 56','santos@gmail.com','086767676767'),
    -> ('50004','dr. Putri, Sp.M','mata','wanita','jl. utan kayu no 21','putri@gmail.com','084545454545'),
    -> ('50005','dr. Anna, Sp.S','saraf','wanita','jl. pemuda no 44','anna@gmail.com','087979797979');
Query OK, 5 rows affected (0.08 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [rumah_sakit]> select * from dokter;
+-----+-----+-----+-----+-----+
| kd_dokter | nama_dokter      | spesialis | jk   | alamat          | em
ail
+-----+-----+-----+-----+-----+
| 50001 | dr. Marini, Sp.A | anak      | wanita | jl. merdeka no 11 | ma
rini@gmail.com | 089090909090 |
| 50002 | dr. Eddy, Sp.P   | paru      | pria   | jl. juang no 87 | ed
dy@gmail.com | 087979797979 |
| 50003 | dr. Santoso, Sp.P | paru      | pria   | jl. kalimalang no 56 | sa
ntoso@gmail.com | 086767676767 |
| 50004 | dr. Putri, Sp.M  | mata     | wanita | jl. utan kayu no 21 | pu
tri@gmail.com | 084545454545 |
| 50005 | dr. Anna, Sp.S   | saraf     | wanita | jl. pemuda no 44 | an
na@gmail.com | 087979797979 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

c) Tabel periksa_pasien

```
MariaDB [rumah_sakit]> insert into periksa_pasien values
    -> ('PP10001','50001','10001','demam','ambil darah','150000'),
    -> ('PP10002','50002','10003','bronkitis','rontgen','300000'),
    -> ('PP10003','50003','10005','tbc','rontgen','300000'),
    -> ('PP10004','50004','10002','minus','laser','10000000'),
    -> ('PP10005','50005','10004','stroke','ct scan','800000');
Query OK, 5 rows affected (0.08 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [rumah_sakit]> select * from periksa;
ERROR 1146 (42S02): Table 'rumah_sakit.periksa' doesn't exist
MariaDB [rumah_sakit]> select * from periksa_pasien;
+-----+-----+-----+-----+-----+
| kd_periksa | kd_dokter | kd_pasien | diagnosa | tindakan      | biaya_tindakan
+-----+-----+-----+-----+-----+
| PP10001   | 50001   | 10001   | demam   | ambil darah  | 150000
| PP10002   | 50002   | 10003   | bronkitis | rontgen     | 300000
| PP10003   | 50003   | 10005   | tbc      | rontgen     | 300000
| PP10004   | 50004   | 10002   | minus    | laser       | 10000000
| PP10005   | 50005   | 10004   | stroke   | ct scan     | 800000
+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)
```

d) Tabel obat

```
MariaDB [rumah_sakit1]> insert into obat values
-> ('OB001','isoniazid','pil','500','2017-03-10','50000'),
-> ('OB002','proris','larutan','100','2018-05-15','70000'),
-> ('OB003','sanmol','larutan','120','2018-11-22','75000'),
-> ('OB004','promato','larutan','240','2018-03-25','55000'),
-> ('OB005','aspirin','pil','450','2018-12-01','65000');
Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [rumah_sakit1]> select * from obat;
+-----+-----+-----+-----+-----+-----+
| kd_obat | nama_obat | jenis | stok | tgl_expired | harga |
+-----+-----+-----+-----+-----+-----+
| OB001 | isoniazid | pil | 500 | 2017-03-10 | 50000 |
| OB002 | proris | larutan | 100 | 2018-05-15 | 70000 |
| OB003 | sanmol | larutan | 120 | 2018-11-22 | 75000 |
| OB004 | promato | larutan | 240 | 2018-03-25 | 55000 |
| OB005 | aspirin | pil | 450 | 2018-12-01 | 65000 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

e) Tabel resep

```
MariaDB [rumah_sakit1]> insert into resep values
-> ('R0001','PP10001','OB002','2016-02-04','3x1','1'),
-> ('R0002','PP10002','OB005','2016-02-04','3x1','2'),
-> ('R0003','PP10003','OB001','2016-02-05','3x1','1'),
-> ('R0004','PP10004','OB004','2016-02-05','4x2','1'),
-> ('R0005','PP10005','OB005','2016-02-05','3x1','2');
Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [rumah_sakit1]> select * from resep;
+-----+-----+-----+-----+-----+-----+
| no_resep | kd_periksa | kd_obat | tgl_resep | aturan_pakai | jumlah_obat |
+-----+-----+-----+-----+-----+-----+
| R0001 | PP10001 | OB002 | 2016-02-04 | 3x1 | 1 |
| R0002 | PP10002 | OB005 | 2016-02-04 | 3x1 | 2 |
| R0003 | PP10003 | OB001 | 2016-02-05 | 3x1 | 1 |
| R0004 | PP10004 | OB004 | 2016-02-05 | 4x2 | 1 |
| R0005 | PP10005 | OB005 | 2016-02-05 | 3x1 | 2 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

2. Perintah update

```
UPDATE table_name SET field_name = 'new_value' WHERE some_field_name = 'some_value';
```

a) Tabel pasien

Update alamat menjadi jl. Markisa no 5 dengan kd_pasien = 10001

```

MariaDB [rumah_sakit]> update pasien set alamat='jl. markisa no 5' where kd_pasien='10001';
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [rumah_sakit]> select * from pasien;
+-----+-----+-----+-----+-----+
| kd_pasien | nama    | jk      | tempat_tinggal | tgl_lahir | alamat
| usia     |
+-----+-----+-----+-----+-----+
| 10001   | ardi    | pria   | jakarta       | 1992-05-02 | jl. markisa no 5
| 23      |
| 10002   | firna   | wanita  | bekasi        | 1995-01-01 | jl. pekan raya no
| 90      | 21      |
| 10003   | jeffrey  | pria   | bandung       | 1995-09-11 | jl. raya cakung n
| 01      | 21      |
| 10004   | glenn   | pria   | jakarta       | 2000-12-12 | jl. tanjungan no
| 13      | 15      |
| 10005   | asri    | wanita  | bekasi        | 1996-11-10 | jl. merdeka no 89
| 20      |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

b) Tabel dokter

Update no_telp menjadi 081230000012 dengan kd_dokter = 50003

```

MariaDB [rumah_sakit]> update dokter set no_telp='081230000012' where kd_dokter='50003';
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [rumah_sakit]> select * from dokter;
+-----+-----+-----+-----+-----+
| kd_dokter | nama_dokter           | spesialis | jk      | alamat
| email     | no_telp                |
+-----+-----+-----+-----+-----+
| 50001   | dr. Marini, Sp.A      | anak     | wanita  | jl. merdeka no 11  | ma
| rini@gmail.com | 089090909090 |
| 50002   | dr. Eddy, Sp.P        | paru     | pria    | jl. juang no 87  | ed
| dy@gmail.com | 087979797979 |
| 50003   | dr. Santoso, Sp.P     | paru     | pria    | jl. kalimalang no 56 | sa
| ntoso@gmail.com | 081230000012 |
| 50004   | dr. Putri, Sp.M       | mata    | wanita  | jl. utan kayu no 21 | pu
| tri@gmail.com | 084545454545 |
| 50005   | dr. Anna, Sp.S        | saraf    | wanita  | jl. pemuda no 44  | an
| na@gmail.com | 087979797979 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

Update email menjadi 'rini1967@gmail.com' dengan kd_dokter = 50001

```
MariaDB [rumah_sakit]> update dokter set email='rini1967@gmail.com' where kd_dokter='50001';
Query OK, 1 row affected <0.09 sec>
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [rumah_sakit]> select * from dokter;
+-----+-----+-----+-----+-----+
| kd_dokter | nama_dokter      | spesialis | jk    | alamat          | em
ail          | no_telp           |           |       |                 | ail
+-----+-----+-----+-----+-----+
| 50001   | dr. Marini, Sp.A | anak      | wanita | jl. merdeka no 11 | ri
nii1967@gmail.com | 089090909090 |
| 50002   | dr. Eddy, Sp.P  | paru      | pria   | jl. juang no 87  | ed
dy@gmail.com     | 087979797979 |
| 50003   | dr. Santoso, Sp.P| paru      | pria   | jl. kalimalang no 56 | sa
ntoso@gmail.com  | 081230000012 |
| 50004   | dr. Putri, Sp.M  | mata     | wanita | jl. utan kayu no 21 | pu
tri@gmail.com   | 084545454545 |
| 50005   | dr. Anna, Sp.S  | saraf     | wanita | jl. pemuda no 44  | an
na@gmail.com    | 087979797979 |
+-----+-----+-----+-----+-----+
5 rows in set <0.00 sec>
```

c) Tabel obat

Update stok menjadi 255 dengan kd_obat ='OB004'

Update stok menjadi 320 dengan kd_obat ='OB005'

```
MariaDB [rumah_sakit]> update obat set stok='255' where kd_obat='OB004';
Query OK, 0 rows affected <0.09 sec>
Rows matched: 1  Changed: 0  Warnings: 0

MariaDB [rumah_sakit]> update obat set stok='320' where kd_obat='OB005';
Query OK, 1 row affected <0.09 sec>
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [rumah_sakit]> select * from obat;
+-----+-----+-----+-----+-----+
| kd_obat | nama_obat | jenis | stok | tgl_expired | harga |
+-----+-----+-----+-----+-----+
| OB001  | isoniazid | pil   | 500  | 2017-03-10  | 50000 |
| OB002  | proris    | larutan | 100  | 2018-05-15  | 70000 |
| OB003  | sanmol    | larutan | 120  | 2018-11-22  | 75000 |
| OB004  | promato   | larutan | 255  | 2018-03-25  | 55000 |
| OB005  | aspirin   | pil   | 320  | 2018-12-01  | 65000 |
+-----+-----+-----+-----+-----+
5 rows in set <0.00 sec>
```

3. Perintah delete

DELETE FROM table_name WHERE 'some_field' = 'some_value';

Contoh:

Hapus record dengan no_resep = 'R0005'

```
MariaDB [rumah_sakit]> delete from resep where no_resep='R0005';
Query OK, 1 row affected <0.09 sec>

MariaDB [rumah_sakit]> select * from resep;
+-----+-----+-----+-----+-----+
| no_resep | kd_periksa | kd_obat | tgl_resep | aturan_pakai | jumlah_obat |
+-----+-----+-----+-----+-----+
| R0001   | PP10001   | OB002   | 2016-02-04 | 3x1        | 1           |
| R0002   | PP10002   | OB005   | 2016-02-04 | 3x1        | 2           |
| R0003   | PP10003   | OB001   | 2016-02-05 | 3x1        | 1           |
| R0004   | PP10004   | OB004   | 2016-02-05 | 4x2        | 1           |
+-----+-----+-----+-----+-----+
4 rows in set <0.00 sec>
```

Tugas (Take Home)

1. Tugas individu dalam bentuk makalah (cover, daftar isi, pembahasan)
2. Menggunakan font Times New Roman size 11
3. Lakukan dokumentasi (screenshoot) pada setiap query dan hasilnya (show tables, desc table, select *)
4. Rancanglah database sebagai berikut:
 - Buatlah database **kuliah**
 - Minimal terdiri dari 3 tabel yang saling berelasi (Primary Key & Foreign Key)
 - Setiap tabel terdiri:
 - Terdapat perintah DML (INSERT (10), SELECT (1), UPDATE (5), DELETE(2))

PERTEMUAN 5

PERINTAH DCL

Data Control Language

Data Control Language (DCL) ialah perintah yang digunakan untuk melakukan pengontrolan data dan server databasenya.

- GRANT – untuk memberikan hak akses pengguna ke database.
- REVOKE – untuk menghilangkan hak akses yang telah diberikan dengan perintah GRANT.

Jenis Hak Akses berdasarkan cakupan level

MySQL menyediakan berbagai tingkatan level hak akses. Setiap user dapat dibatasi untuk dapat mengakses baik itu sebuah database tertentu saja, tabel tertentu, atau bahkan hanya kolom tertentu.

Jika didasarkan pada pengelompokan ini, kita dapat membagi hak akses MySQL menjadi 4 level tingkatan, yaitu:

1. Hak akses global (*.*)

Hak akses ini berarti user dapat memiliki hak akses untuk seluruh database yang terdapat di dalam MySQL.

Sintaks:

GRANT SELECT ON *.* TO 'user'@'localhost';

Perhatikan cara penulisan nama_database.nama_tabel, dimana kita menulisnya dengan *.*, sehingga user tersebut dapat mengakses seluruh tabel pada seluruh database.

2. Hak Akses Level Database (nama_database.*)

Hak akses ini berarti user memiliki hak akses penuh untuk sebuah database.

Sintaks:

GRANT SELECT ON nama_database.* TO 'user'@'localhost';

Untuk penulisan nama_database.nama_tabel, kita membatasi nama database, namun memberikan hak akses untuk seluruh tabel, penulisannya adalah nama_database.*

3. Hak Akses Level Tabel (nama_database.nama_tabel)

Hak akses ini berarti user memiliki hak akses untuk sebuah tabel yang berada pada sebuah

database.

Sintaks:

GRANT SELECT ON nama_database.nama_tabel TO 'user'@'localhost';

Hak akses yang dimiliki user hanya terbatas pada level sebuah tabel saja.

4. Hak Akses Level Kolom (nama_kolom)

Hak akses ini adalah hak akses paling kecil yang dapat diberikan kepada sebuah user. Dengan hak akses level kolom, user hanya memiliki hak akses untuk beberapa kolom pada sebuah tabel.

Sintaks:

GRANT SELECT (kolom1,kolom2) ON nama_database.nama_tabel TO 'user'@'localhost';
Level paling akhir ini kita membatasi hak akses user hanya untuk kolom tertentu saja. Penulisan kolom yang diperbolehkan diletakkan di dalam tanda kurung.

Latihan:

Sebelum memasuki kontrol aksi user, buatlah 3 admin sebagai berikut:

```
CREATE USER 'nama_user'@'localhost';
```

- admin1@localhost
- dokter@localhost
- pasien@localhost

Tampilkan user yang ada pada database mysql

```
MariaDB [mysql]> select user,host from mysql.user;
+-----+-----+
| user | host  |
+-----+-----+
| root | 127.0.0.1
| root | ::1
|      | localhost
| admin1| localhost
| dokter| localhost
| pasien| localhost
| pma   | localhost
| root  | localhost
+-----+-----+
8 rows in set (0.00 sec)
```

1. Memberikan Hak Akses Kepada User dengan query GRANT

Grant berfungsi untuk membuat user baru dan memberikan hak istimewa. Grant adalah salah satu privilege untuk tabel. Grant digunakan untuk memberikan privilege kepada tabel yang didefinisikan kepada pemakai lain. Privilege untuk pemakai dalam perintah grant didefinisikan dengan menggunakan nama-nama privilege. Nama privilege memudahkan administrator untuk dapat memberikan privilege tanpa harus tahu apa nama field dan tabel yang harus diisi.

Sintaks:

```
GRANT hak_akses ON nama_database.nama_tabel TO 'nama_user'@'lokasi_user';
```

Contoh: grant select on rumah_sakit.* to admin1@localhost

```
MariaDB [rumah_sakit]> grant select on rumah_sakit.* to admin1@localhost;
Query OK, 0 rows affected (0.06 sec)
```

Lalu masuk sebagai admin1 dan tampilkan database

```
C:\xampp\mysql\bin>mysql -u admin1
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 33
Server version: 10.1.10-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [none]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| rumah_sakit |
| test |
+-----+
3 rows in set (0.00 sec)
```

Tampilkan tabel

```
MariaDB [rumah_sakit]> show tables;
+-----+
| Tables_in_rumah_sakit |
+-----+
| dokter |
| obat |
| pasien |
| periksa_pasien |
| resep |
+-----+
5 rows in set (0.00 sec)
```

Gunakan database rumah_sakit dan masukkan sintaks select tabel pasien

```
MariaDB [rumah_sakit]> select * from pasien;
+-----+
| kd_pasien | nama    | jk      | tempat_tinggal | tgl_lahir | alamat
| usia     |
+-----+
| 10001   | ardi    | pria   | jakarta       | 1992-05-02 | jl. markisa no 5
| 23      |
| 10002   | firna   | wanita | bekasi        | 1995-01-01 | jl. pekan raya no
| 90      | 21      |
| 10003   | jeffrey | pria   | bandung       | 1995-09-11 | jl. raya cakung n
| o 1    | 21      |
| 10004   | glenn   | pria   | jakarta       | 2000-12-12 | jl. tanjungan no
| 13      | 15      |
| 10005   | asri    | wanita | bekasi        | 1996-11-10 | jl. merdeka no 89
| 20      |
+-----+
5 rows in set (0.00 sec)
```

Coba lakukan query update dan drop

```
MariaDB [rumah_sakit]> update pasien set jk='wanita' where nama='ardi';
ERROR 1142 (42000): UPDATE command denied to user 'admin1'@'localhost' for table
'pasien'
MariaDB [rumah_sakit]> drop table pasien;
ERROR 1142 (42000): DROP command denied to user 'admin1'@'localhost' for table
'pasien'
MariaDB [rumah_sakit]>
```

Dari contoh query diatas, dapat dilihat bahwa pada saat admin1 menjalankan perintah SHOW TABLES, ia hanya dapat melihat satu tabel, yakni tabel pasien. Padahal dalam database tersebut

kita juga telah membuat tabel dokter, periksa, obat dan resep, namun karena hak akses yang diberikan, admin1 hanya dapat melihat tabel yang diperbolehkan yaitu tabel pasien

Setelah menampilkan isi tabel pasien, user admin1 mencoba menghapus tabel pasien. Tetapi karena kita membatasi hak aksesnya, admin1 tidak dapat menjalankan query UPDATE dan DROP, dan akan langsung ditolak oleh MySQL..

Memberikan Seluruh Hak Akses (query GRANT ALL)

GRANT ALL adalah cara peningkatan memberikan hampir semua hak akses kepada sebuah user tertentu. Hak akses ini mencakup seluruh query dasar:

ALTER, CREATE, CREATE TEMPORARY TABLES, DELETE, DROP, EXECUTE, FILE, INDEX, INSERT, LOCK TABLES, PROCESS, RELOAD, REPLICATION CLIENT, REPLICATION SLAVE, SELECT, SHOW DATABASES, SHUTDOWN, SUPER, dan UPDATE.

Dokter diberi hak akses secara keseluruhan dan dapat melakukan manipulasi data pada tabel resep.

```
MariaDB [mysql]> grant all on rumah_sakit.resep to dokter@localhost;
Query OK, 0 rows affected (0.00 sec)
```

Dengan memberikan hak akses GRANT ALL, maka user dokter dapat menggunakan seluruh query dasar pada tabel resep, seperti SELECT, UPDATE, bahkan DELETE. Sebagai latihan, silahkan mencoba masuk sebagai user dokter dan lakukan perintah seperti UPDATE, DELETE, dan DROP.

Memberikan Hak Akses MySQL Pada Level Kolom

Pasien diberi hak akses untuk melihat tabel resep dengan kolom no_resep, kd_obat, aturan_pakai dan jumlah_obat.

```
MariaDB [mysql]> grant select (no_resep, kd_obat, aturan_pakai, jumlah_obat) on
rumah_sakit.resep to pasien@localhost;
Query OK, 0 rows affected (0.01 sec)
```

Untuk mengujinya, masuklah sebagai user pasien:

```

C:\xampp\mysql\bin>mysql -u pasien
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
Server version: 10.1.10-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> use rumah_sakit;
Database changed
MariaDB [rumah_sakit]> show tables;
+-----+
| Tables_in_rumah_sakit |
+-----+
| resep                |
+-----+
1 row in set (0.00 sec)

MariaDB [rumah_sakit]> select * from pasien;
ERROR 1142 (42000): SELECT command denied to user 'pasien'@'localhost' for table
'pasien'
MariaDB [rumah_sakit]> select no_resep,kd_obat,aturan_pakai,jumlah_obat from res
ep;
+-----+-----+-----+-----+
| no_resep | kd_obat | aturan_pakai | jumlah_obat |
+-----+-----+-----+-----+
| R0001   | OB002  | 3x1       | 1           |
| R0002   | OB005  | 3x1       | 2           |
| R0003   | OB001  | 3x1       | 1           |
| R0004   | OB004  | 4x2       | 1           |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

Pada saat user pasien mencoba menampilkan seluruh kolom dengan query `SELECT * FROM resep`, MySQL akan mengeluarkan error karena user pasien hanya memiliki hak akses untuk kolom `no_resep`, `kd_obat`, `aturan_pakai` dan `jumlah_obat` saja, dimana pada saat menggunakan perintah `SELECT no_resep, kd_obat, aturan_pakai dan jumlah_obat FROM resep`, MySQL menampilkannya dengan baik.

Cara Melihat Daftar User MySQL

Untuk melihat user yang terdaftar di dalam MySQL Server, kita dapat mengaksesnya dengan melihat tabel user yang terdapat di dalam database `mysql`. Database `mysql` terdiri dari banyak tabel, namun untuk keperluan user, kita hanya akan menggunakan tabel `user`. Berikut query untuk melihat seluruh user yang terdaftar dalam MySQL:

```
SELECT user,host FROM mysql.user;
```

Melihat Hak Akses User MySQL (SHOW GRANTS FOR)

Format dasar query `SHOW GRANTS FOR` adalah sebagai berikut:

```
SHOW GRANTS FOR 'nama_user'@'lokasi_user';
```

```

MariaDB [rumah_sakit]> show grants for admin1@localhost;
+-----+
| Grants for admin1@localhost |
+-----+
| GRANT USAGE ON *.* TO 'admin1'@'localhost' |
| GRANT SELECT ON `rumah_sakit`.* TO 'admin1'@'localhost' |
+-----+
2 rows in set (0.00 sec)

```

2. Perintah Revoke

Hak akses yang diberikan ke seorang user, adakalanya perlu dilakukan perubahan, tergantung kondisi dan kebijakan pengguna. Untuk menghapus user, kita dapat menggunakan query DROP user, namun kadang kita perlu hanya menghapus hak aksesnya saja, tanpa menghapus user yang bersangkutan. Untuk hal ini MySQL menyediakan perintah REVOKE.

Sintaks:

```
REVOKE jenis_hak_akses (kolom1,kolom2) ONnama_database.nama_tabel FROM  
nama_user@lokasi_user;
```

- **Jenis_hak_akses** adalah privileges yang akan dihapus dari user tersebut. Kita bisa membuat hak akses secara satu persatu untuk keperluan yang spesifik, atau mengisikan perintah **ALL PRIVILEGES** untuk menghapus seluruh hak akses.
- **kolom1,kolom2** adalah nama judul kolom yang hak aksesnya akan dicabut. Jika tidak diisi, maka dianggap query **REVOKE** akan menghapus pada seluruh kolom.
- **nama_database** adalah nama database yang ingin dihapuskan hak aksesnya.**nama_database** bisa ditulis dengan tanda bintang (*) untuk merujuk kepada seluruh database.
- **nama_tabel** adalah nama tabel yang ingin dihapuskan hak aksesnya. **nama_tabel** bisa ditulis dengan tanda bintang (*) untuk merujuk kepada seluruh tabel.
- **nama_user** adalah nama dari user yang akan dihapus hak aksesnya.
- **lokasi_user** adalah alamat IP dari user yang akan dihapus hak aksesnya.

Sebagai contoh, kita akan menghapus hak akses **SELECT** dari user **admin1**:

Masuk sebagai user root dan gunakan database mysql

```
MariaDB [<none>]> show grants for admin1@localhost;  
+-----+  
| Grants for admin1@localhost |  
+-----+  
| GRANT USAGE ON *.* TO 'admin1'@'localhost'  
| GRANT SELECT ON `rumah_sakit`.* TO 'admin1'@'localhost'  
+-----+  
2 rows in set (0.00 sec)  
  
MariaDB [<none>]> revoke select on rumah_sakit.* from admin1@localhost;  
Query OK, 0 rows affected (0.00 sec)  
  
MariaDB [<none>]> show grants for admin1@localhost;  
+-----+  
| Grants for admin1@localhost |  
+-----+  
| GRANT USAGE ON *.* TO 'admin1'@'localhost' |  
+-----+  
1 row in set (0.00 sec)
```

Untuk menguji bahwa user **admin1** sudah tidak dapat melihat database **rumah_sakit** lagi, saya akan kembali masuk sebagai user **admin1**:

```
MariaDB [<none>]> use rumah_sakit;  
ERROR 1044 (42000): Access denied for user ''@'localhost' to database 'rumah_sakit'
```

Database **rumah_sakit** tidak dapat digunakan karena hak akses sudah dibatalkan (revoke).

Tugas

GRANT

Admin

1. Hak akses level update tabel database (seluruh tabel)

Dokter

2. Hak akses update level tabel periksa_pasien
3. Hak akses select level tabel pasien
4. Hak akses update level tabel resep

Pasien

5. Hak akses select level kolom tabel obat (nama obat dan harga)
6. Hak akses update level kolom tabel dokter (kode dokter dan nama dokter)

REVOKE

Pasien

7. Batalkan hak akses level tabel dokter

PERTEMUAN 6

BAHASA QUERY FORMAL PROSEDURAL

Bahasa Query Formal Prosedural

Definisi: Bahasa query yang diterjemahkan dengan menggunakan simbol-simbol matematis dengan memberi spesifikasi data apa yang dibutuhkan dan bagaimana cara mendapatkannya.

1. Perintah select

- **Menampilkan Seluruh Isi Tabel MySQL**

Format dasar query select untuk menampilkan seluruh isi tabel adalah sebagai berikut:

```
SELECT * FROM nama_tabel
```

Tanda bintang (*) adalah wildcard sebagai pengganti ‘pilih semua kolom’.

Sebagai contoh, berikut query untuk menampilkan seluruh isi tabel dokter:

```
MariaDB [rumah_sakit]> select * from dokter;
```

kd_dokter	nama_dokter	spesialis	jk	alamat	email	no_telp
50001	dr. Marini, Sp.A	anak	wanita	jl. merdeka no 11	rini1967@gmail.com	089090909090
50002	dr. Eddy, Sp.P	paru	pria	jl. juang no 87	eddy@gmail.com	087979797979
50003	dr. Santoso, Sp.P	paru	pria	jl. kalimalang no 56	santoso@gmail.com	081230000012
50004	dr. Putri, Sp.M	mata	wanita	jl. utan kayu no 21	putri@gmail.com	084545454545
50005	dr. Anna, Sp.S	saraf	wanita	jl. pemuda no 44	anna@gmail.com	087979797979

5 rows in set (0.00 sec)

- **Menampilkan Kolom Tertentu dari Tabel MySQL (SELECT ... FROM)**

Sintaks:

```
SELECT nama_kolom1, nama_kolom2,... FROM nama_tabel;
```

nama_kolom1 dan nama_kolom2 adalah nama kolom yang ingin kita tampilkan. Misalkan kita ingin menampilkan kolom nama_dokter dan alamat dari tabel dokter, maka querynya adalah sebagai berikut:

```
MariaDB [rumah_sakit]> select nama_dokter,alamat from dokter;
```

nama_dokter	alamat
dr. Marini, Sp.A	jl. merdeka no 11
dr. Eddy, Sp.P	jl. juang no 87
dr. Santoso, Sp.P	jl. kalimalang no 56
dr. Putri, Sp.M	jl. utan kayu no 21
dr. Anna, Sp.S	jl. pemuda no 44

5 rows in set (0.00 sec)

- **Menfilter/Menyeleksi data dari Tabel MySQL (SELECT...WHERE...)**

Kondisi WHERE pada perintah SELECT digunakan untuk menyeleksi data yang diinginkan, sedangkan data yang tidak memenuhi kriteria, tidak akan ditampilkan.

Format dasar query SELECT...WHERE adalah:

```
SELECT nama_kolom1, nama_kolom2,... FROM nama_tabel WHERE kondisi;
```

Contohnya untuk menampilkan data dokter yang beralamat di jl pemuda no 44, dapat menggunakan query berikut:

```
MariaDB [rumah_sakit]> select kd_dokter, nama_dokter, no_telp from dokter where alamat ='jl. pemuda no 44';
+-----+-----+-----+
| kd_dokter | nama_dokter | no_telp |
+-----+-----+-----+
| 50005 | dr. Anna, Sp.S | 087979797979 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Kondisi WHERE sangat fleksibel dan kita juga bisa menggunakan kondisi operasi slain seperti lebih besar (>), lebih kecil (<), tidak sama (<>), dan lain-lain.

List lengkap dari penggunaan kondisi yang dapat digunakan pada SELECT...WHERE adalah:

Operator Logika

Operator	Penjelasan
Not (!)	Bukan
AND (&&)	Dan
OR ()	Atau

Operator Aritmatika

Operator	Penjelasan
+ , -	Tambah, Kurang
* , /	Kali, Bagi
%	Modulus

Mengurutkan hasil tampilan data MySQL (SELECT...ORDER BY)

MySQL menyediakan perintah opsional ORDER BY untuk mengurutkan data yang dihasilkan. Query dasar untuk SELECT...ORDER BY adalah:

```
SELECT nama_kolom1,... FROM nama_tabel WHERE kondisi ORDER BY nama_kolom_urut;
```

nama_kolom_urut adalah kolom yang akan kita urutkan. Pengurutan bisa dari paling kecil ke besar, ataupun besar ke kecil. Pilihan ini dapat diatur dengan penambahan instruksi ASC (singkatan dari ascending) untuk pengurutan dari kecil ke besar, dan DESC (singkatan dari descending) untuk urutan dari besar ke kecil. Jika tidak dijelaskan, secara default bawaan MySQL perintah ORDER BY akan menggunakan ASC.

Jika kita ingin menampilkan seluruh dokter pada tabel dokter dan diurutkan kolom alamat secara abjad, maka querynya adalah sebagai berikut:

```
MariaDB [rumah_sakit]> select * from dokter order by nama_dokter asc;
+-----+-----+-----+-----+-----+-----+
| kd_dokter | nama_dokter | spesialis | jk | alamat | email | no_telp |
+-----+-----+-----+-----+-----+-----+
| 50005 | dr. Anna, Sp.S | saraf | wanita | jl. pemuda no 44 | anna@gmail.com | 087979797979 |
| 50002 | dr. Eddy, Sp.P | paru | pria | jl. juang no 87 | eddy@gmail.com | 087979797979 |
| 50001 | dr. Marini, Sp.A | anak | wanita | jl. merdeka no 11 | rini1967@gmail.com | 089090909090 |
| 50004 | dr. Putri, Sp.M | mata | wanita | jl. utan kayu no 21 | putri@gmail.com | 084545454545 |
| 50003 | dr. Santoso, Sp.P | paru | pria | jl. kalimalang no 56 | santoso@gmail.com | 081230000012 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

• Membatasi Hasil query SELECT (SELECT...LIMIT)

MySQL menyediakan pilihan opsional LIMIT untuk membatasi hasil query SELECT, format dasar query SELECT...LIMIT adalah sebagai berikut:

```
SELECT nama_kolom1 FROM nama_tabel WHERE kondisi LIMIT baris_awal, jumlah_baris;
```

Dimana baris_awal adalah awal nomor baris yang ingin ditampilkan, dan jumlah_baris adalah jumlah baris yang diurutkan dari baris_awal. Nomor baris pada MySQL diawali dengan nomor 0.

Misalkan kita ingin menampilkan data 3 nama dokter paling atas yang dirutkan berdasarkan nama, maka querynya adalah:

```
MariaDB [rumah_sakit]> select * from dokter order by nama_dokter asc limit 0,3;
```

kd_dokter	nama_dokter	spesialis	jk	alamat	email	no_telp
50005	dr. Anna, Sp.S	saraf	wanita	jl. pemuda no 44	anna@gmail.com	087979797979
50002	dr. Eddy, Sp.P	paru	pria	jl. juang no 87	eddy@gmail.com	087979797979
50001	dr. Marini, Sp.A	anak	wanita	jl. merdeka no 11	rini1967@gmail.com	089090909090

3 rows in set (0.00 sec)

- **Pencarian Data Tabel MySQL Menggunakan query SELECT..LIKE**

Sintaks:

```
SELECT  nama_kolom_tampil   FROM   nama_tabel   WHERE   nama_kolom_cari   LIKE keyword_pencarian
```

Sebagai contoh awal, misalkan kita ingin menampilkan seluruh kolom dari tabel pasien dimana nama dosen adalah jeffrey. Maka contoh querynya:

```
MariaDB [rumah_sakit]> select * from pasien where nama like 'jeffrey';
```

kd_pasien	nama	jk	tempat_tinggal	tgl_lahir	alamat	usia
10003	jeffrey	pria	bandung	1995-09-11	jln. raya cakung no 1	21

1 row in set (0.00 sec)

Namun katakan kita ingin menampilkan seluruh kolom dari tabel **pasien** dimana **nama pasien** diawali dengan huruf ‘a’. Untuk pencarian seperti inilah kita menggunakan *query LIKE*.

MySQL menyediakan 2 karakter khusus untuk pencarian **LIKE**, yaitu karakter ‘_’ dan ‘%’, penjelasannya:

- ‘_’ : karakter ganti yang cocok untuk satu karakter apa saja.
- ‘%’ : karakter ganti yang cocok untuk karakter apa saja dengan panjang karakter tidak terbatas, termasuk tidak ada karakter.
- Agar mudah memahami, langsung saja kita gunakan untuk pencarian **nama pasien** yang diawali dengan huruf ‘a’:

```
MariaDB [rumah_sakit]> select * from pasien where nama like'a%';
```

kd_pasien	nama	jk	tempat_tinggal	tgl_lahir	alamat	usia
10001	ardi	pria	jakarta	1992-05-02	jln. markisa no 5	23
10005	asri	wanita	bekasi	1996-11-10	jln. merdeka no 89	20

2 rows in set (0.00 sec)

Menggunakan ‘a%’ karena kita ingin mencari **nama pasien** yang diawali dengan a, dan diikuti oleh karakter apa saja dengan panjang tidak dibatasi.

Kita dapat mengganti kata kunci hasil pencarian tersebut dengan karakter lain, sebagai contoh:

- ‘S%’ : Cocok dengan kata yang diawali dengan S, dan diikuti dengan karakter apa saja, contoh: ‘S’, ‘Sa’, ‘Si’, ‘Saaaaaa’, ‘Susi’, dan ‘Sabrina Sari’.
- ‘S_’ : Cocok dengan kata yang diawali dengan S, dan diikuti dengan satu karakter apa saja, contoh: ‘Si’, ‘Sa’, ‘Su’, ‘Se’, dan ‘St’.
- ‘A_i’ : Cocok dengan kata yang diawali dengan ‘A’, diikuti oleh 2 karakter bebas, namun diakhiri dengan i, contoh: ‘Andi’, ‘ardi’, ‘aaai’.
- ‘%e’ : Cocok dengan seluruh kata dengan panjang berapapun yang diakhiri dengan huruf ‘e’, contoh: ‘Kece’, ‘Kue’, dan ‘mie’.

- ‘%dia%’: Cocok dengan seluruh kata yang mengandung kata ‘dia’, contoh: ‘media’, ‘kemudian’, ‘dia’, dan ‘diaaaa’.

2. Perintah where

- **Select**

Misalkan kita ingin menampilkan data dokter dengan kondisi tempat_tinggal berada di jakarta, maka querynya adalah:

```
MariaDB [rumah_sakit]> select * from pasien where tempat_tinggal='jakarta';
```

kd_pasien	nama	jk	tempat_tinggal	tgl_lahir	alamat	usia
10001	ardi	pria	jakarta	1992-05-02	jl. markisa no 5	23
10004	glenn	pria	jakarta	2000-12-12	jl. tanjungan no 13	15

2 rows in set (0.00 sec)

- **Update**

Misalkan kita ingin mengupdate data alamat dokter dengan kondisi kd_dokter = 50001, maka querynya adalah:

```
MariaDB [rumah_sakit]> select * from dokter;
```

kd_dokter	nama_dokter	spesialis	jk	alamat	email	no_telp
50001	dr. Marini, Sp.A	anak	wanita	jl. merdeka no 11	rini1967@gmail.com	089090909090
50002	dr. Eddy, Sp.P	paru	pria	jl. juang no 87	eddy@gmail.com	087979797979
50003	dr. Santoso, Sp.P	paru	pria	jl. kalimalang no 56	santoso@gmail.com	081230000012
50004	dr. Putri, Sp.M	mata	wanita	jl. utan kayu no 21	putri@gmail.com	084545454545
50005	dr. Anna, Sp.S	saraf	wanita	jl. pemuda no 44	anna@gmail.com	087979797979

5 rows in set (0.00 sec)

```
MariaDB [rumah_sakit]> update dokter set alamat='jl. sudirman no 50' where kd_dokter='50001';
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
MariaDB [rumah_sakit]> select * from dokter;
```

kd_dokter	nama_dokter	spesialis	jk	alamat	email	no_telp
50001	dr. Marini, Sp.A	anak	wanita	jl. sudirman no 50	rini1967@gmail.com	089090909090
50002	dr. Eddy, Sp.P	paru	pria	jl. juang no 87	eddy@gmail.com	087979797979
50003	dr. Santoso, Sp.P	paru	pria	jl. kalimalang no 56	santoso@gmail.com	081230000012
50004	dr. Putri, Sp.M	mata	wanita	jl. utan kayu no 21	putri@gmail.com	084545454545
50005	dr. Anna, Sp.S	saraf	wanita	jl. pemuda no 44	anna@gmail.com	087979797979

5 rows in set (0.00 sec)

- **Delete**

Misalkan kita ingin menghapus data periksa_pasien dengan kondisi kd_periksa = PP10005, maka querynya adalah:

```
MariaDB [rumah_sakit]> select * from periksa_pasien;
+-----+-----+-----+-----+-----+-----+
| kd_periksa | kd_dokter | kd_pasien | diagnosa | tindakan | biaya_tindakan |
+-----+-----+-----+-----+-----+-----+
| PP10001 | 50001 | 10001 | demam | ambil darah | 150000 |
| PP10002 | 50002 | 10003 | bronkitis | rontgen | 300000 |
| PP10003 | 50003 | 10005 | tbc | rontgen | 300000 |
| PP10004 | 50004 | 10002 | minus | laser | 10000000 |
| PP10005 | 50005 | 10004 | stroke | ct scan | 800000 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.04 sec)

MariaDB [rumah_sakit]> delete from periksa_pasien where kd_periksa='PP10005';
Query OK, 1 row affected (0.11 sec)

MariaDB [rumah_sakit]> select * from periksa_pasien;
+-----+-----+-----+-----+-----+-----+
| kd_periksa | kd_dokter | kd_pasien | diagnosa | tindakan | biaya_tindakan |
+-----+-----+-----+-----+-----+-----+
| PP10001 | 50001 | 10001 | demam | ambil darah | 150000 |
| PP10002 | 50002 | 10003 | bronkitis | rontgen | 300000 |
| PP10003 | 50003 | 10005 | tbc | rontgen | 300000 |
| PP10004 | 50004 | 10002 | minus | laser | 10000000 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

3. Operator Relasional

Operasi Perbandingan

Operator	Penjelasan
=	Sama Dengan
<> atau !=	Tidak Sama Dengan
<	Kurang Dari
<=	Kurang dari atau Sama Dengan
>	Lebih Besar Dari
>=	Lebih Besar Dari atau Sama Dengan
Between	Berada pada batas tertentu
IN	Berada di Dalam
Is Null	Pengecekan apakah berisi NULL
Is Not Null	Pengecekan apakah bukan berisi NULL
Like	Pencarian menggunakan wildcard

Contohnya jika kita menginginkan tampilan tabel obat dengan jumlah stok lebih dari 300 secara berurutan dari yang paling kecil, maka querynya:

```
MariaDB [rumah_sakit]> select * from obat where stok >=150 order by stok;
+-----+-----+-----+-----+-----+
| kd_obat | nama_obat | jenis | stok | tgl_expired | harga |
+-----+-----+-----+-----+-----+
| OB004 | promato | larutan | 255 | 2018-03-25 | 55000 |
| OB005 | aspirin | pil | 320 | 2018-12-01 | 65000 |
| OB001 | isoniazid | pil | 500 | 2017-03-10 | 50000 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

4. Operator AND, OR dan NOT

- AND

Operator adn berfungsi untuk mencari dan menampilkan data yang lebih kurat. Dengan operator AND akan ditampilkan data yang hanya memenuhi kedua syarat yang ditentukan.

Sintaks :

```
SELECT * FROM [NAMA TABEL] WHERE [NAMA KOLOM] = '[PENCARIAN 1]' AND [NAMA KOLOM] = '[PENCARIAN 2]';
```

```
MariaDB [rumah_sakit]> select * from pasien where nama = 'ardi' and tempat_tinggal = 'jakarta';
+-----+-----+-----+-----+-----+-----+
| kd_pasien | nama | jk | tempat_tinggal | tgl_lahir | alamat | usia |
+-----+-----+-----+-----+-----+-----+
| 10001 | ardi | pria | jakarta | 1992-05-02 | jl. markisa no 5 | 23 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

- OR

Fungsi operator OR mirip dengan operator AND, namun jika operator AND menampilkan data yang harusmemenuhi keduasyarat yang dibutuhkan, operator OR akan menampilkan data yang hanya memenuh isalah satu dari kedua syarat yang ditentukan.

Sintaks :

```
SELECT * FROM [NAMA TABEL] WHERE [NAMA KOLOM] = '[PENCARIAN 1]' OR [NAMA KOLOM] = '[PENCARIAN 2]';
```

```
MariaDB [rumah_sakit]> select * from pasien where nama = 'ardi' or tempat_tinggal = 'bekasi';
+-----+-----+-----+-----+-----+-----+
| kd_pasien | nama | jk | tempat_tinggal | tgl_lahir | alamat | usia |
+-----+-----+-----+-----+-----+-----+
| 10001 | ardi | pria | jakarta | 1992-05-02 | jl. markisa no 5 | 23 |
| 10002 | firna | wanita | bekasi | 1995-01-01 | jl. pekan raya no 90 | 21 |
| 10005 | asri | wanita | bekasi | 1996-11-10 | jl. merdeka no 89 | 20 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- NOT

Fungsi operator NOT adalah menampilkan data yang tidak sesuai dengan kondisi yang ditentukan.

Sintaks :

```
SELECT * FROM [NAMA TABEL] WHERE NOT [NAMA KOLOM] = '[PENCARIAN]';
```

```
MariaDB [rumah_sakit]> select * from pasien where not tempat_tinggal = 'bekasi';
+-----+-----+-----+-----+-----+-----+
| kd_pasien | nama | jk | tempat_tinggal | tgl_lahir | alamat | usia |
+-----+-----+-----+-----+-----+-----+
| 10001 | ardi | pria | jakarta | 1992-05-02 | jl. markisa no 5 | 23 |
| 10003 | jeffrey | pria | bandung | 1995-09-11 | jl. raya cakung no 1 | 21 |
| 10004 | glenn | pria | jakarta | 2000-12-12 | jl. tanjungan no 13 | 15 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

5. Distinct

Fungsi dari distinct adalah untuk menampilkan hasil query mysql jika ada row yang isinya sama, maka hanya akan diambil salah satu nya saja.

Sintaks:

```
SELECT DISTINCT nama_kolom FROM nama_tabel;
```

- Nama_kolom adalah nama kolom yang akan ditampilkan.
- Nama_tabel adalah nama tabel untuk kolom yang akan ditampilkan.

Sehingga untuk contoh kasus kita sebelumnya untuk menampilkan seluruh tempat_tinggal 1 kali saja, maka querinya:

```
MariaDB [rumah_sakit]> select * from pasien;
+-----+-----+-----+-----+-----+-----+-----+
| kd_pasien | nama | jk | tempat_tinggal | tgl_lahir | alamat | usia |
+-----+-----+-----+-----+-----+-----+-----+
| 10001 | ardi | pria | jakarta | 1992-05-02 | jl. markisa no 5 | 23 |
| 10002 | firna | wanita | bekasi | 1995-01-01 | jl. pekan raya no 90 | 21 |
| 10003 | jeffrey | pria | bandung | 1995-09-11 | jl. raya cakung no 1 | 21 |
| 10004 | glenn | pria | jakarta | 2000-12-12 | jl. tanjungan no 13 | 15 |
| 10005 | asri | wanita | bekasi | 1996-11-10 | jl. merdeka no 89 | 20 |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

MariaDB [rumah_sakit]> select distinct tempat_tinggal from pasien;
+-----+
| tempat_tinggal |
+-----+
| jakarta |
| bekasi |
| bandung |
+-----+
3 rows in set (0.00 sec)
```

Dengan penambahan perintah DISTINCT di awal query SELECT, maka hanya data yang unik saja (data yang tidak sama) yang akan ditampilkan. Seandainya hasil query terdapat data yang sama lebih dari 1 kali tampil, perintah DISTINCT hanya akan menampilkannya 1 kali saja.

Namun jika kita menambahkan kolom nama seperti query berikut:

```
MariaDB [rumah_sakit]> select distinct nama, tempat_tinggal from pasien order by tempat_tinggal;
+-----+-----+
| nama | tempat_tinggal |
+-----+-----+
| jeffrey | bandung |
| asri | bekasi |
| firna | bekasi |
| ardi | jakarta |
| glenn | jakarta |
+-----+-----+
5 rows in set (0.00 sec)
```

Terlihat bahwa MySQL tetap menampilkan seluruh isi tabel tanpa ada yang dieliminasi. Hal ini dikarenakan query DISTINCT hanya mengeleminasi query yang unik, atau tidak sama dilihat secara baris per baris (per record). Dengan mengkombinasikan nama dengan alamat, maka setiap baris dianggap unik, kecuali terdapat nama pasien dan alamat yang persis sama.

6. Between

Operator Between merupakan operator yang digunakan untuk menangani operasi jangkauan.

Sintaks:

```
Select * From Nama_Tabel Where Nama_Field Between 'Ketentuan1' And 'Ketentuan2';
```

```
MariaDB [rumah_sakit]> select * from pasien where usia between '21' and '23';
+-----+-----+-----+-----+-----+-----+-----+
| kd_pasien | nama | jk | tempat_tinggal | tgl_lahir | alamat | usia |
+-----+-----+-----+-----+-----+-----+-----+
| 10001 | ardi | pria | jakarta | 1992-05-02 | jl. markisa no 5 | 23 |
| 10002 | firna | wanita | bekasi | 1995-01-01 | jl. pekan raya no 90 | 21 |
| 10003 | jeffrey | pria | bandung | 1995-09-11 | jl. raya cakung no 1 | 21 |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

7. In

Operator In merupakan operator yang digunakan untuk mencocokkan suatu nilai.

Sintaks:

```
Select Nama_Field From Nama_Tabel Where Nama_Field_Pencocok In ('nilai_field1', 'nilai_field2');
```

```
MariaDB [rumah_sakit]> select kd_pasien, nama, alamat from pasien where nama in ('jeffrey');
+-----+-----+-----+
| kd_pasien | nama      | alamat    |
+-----+-----+-----+
| 10003   | jeffrey   | jl. raya cakung no 1 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

8. As

MySQL menyediakan perintah tambahan AS untuk mengganti sementara nama tabel atau kolom. Disebut sementara karena pada dasarnya nama tabel tersebut tidak berubah, hanya di ganti pada saat ditampilkan dengan query SELECT. Di dalam bahasa SQL, query AS ini lebih dikenal sebagai alias dari nama tabel yang sebenarnya. Alias ditujukan untuk mempermudah penulisan query atau mempercantik tampilan hasil query.

Sintaks:

SELECT, UPDATE, maupun DELETE. AS digunakan sebagai penambahan untuk query SQL lainnya.

Berikut adalah format dasar penulisan alias tabel:

...nama_tabel_asli AS nama_tabel_alias...

nama_tabel_asli adalah nama tabel sesungguhnya yang kita gunakan pada saat pembuatan tabel.

Contohnya adalah tabel dokter dan pasien.

Nama_tabel_alias adalah alias atau nama lain tabel yang ingin kita gunakan, misalnya tabel dokter kita aliaskan menjadi dok saja.

```
MariaDB [rumah_sakit]> select dok.nama_dokter, dok.spesialis, dok.no_telp from dokter as dok;
+-----+-----+-----+
| nama_dokter | spesialis | no_telp |
+-----+-----+-----+
| dr. Marini, Sp.A | anak | 089090909090 |
| dr. Eddy, Sp.P | paru | 087979797979 |
| dr. Santoso, Sp.P | paru | 081230000012 |
| dr. Putri, Sp.M | mata | 084545454545 |
| dr. Anna, Sp.S | saraf | 087979797979 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Dalam query SELECT diatas, saya menampilkan kolom nama_dokter, spesialis dan no_hp dari tabel dokter. Perhatikan di akhir query saya menambahkan perintah dokter AS dok, dalam perintah inilah nama tabel dokter kita aliaskan menjadi dok. Lalu dengan format penulisan nama_tabel.nama_kolom, querynya menjadi SELECT dok.nama_dokter, dok.spesialis, dok.no_hp

Menggunakan ALIAS untuk Mengganti Nama Kolom Tabel MySQL

Selain untuk mengubah nama tabel, MySQL juga menyediakan ALIAS untuk nama kolom. Salah satu kegunaan alias kolom ini adalah untuk merubah nama kolom agar lebih cantik.

Sama seperti ALIAS untuk nama tabel, alias untuk nama kolom juga merupakan perintah tambahan dari query inti seperti SELECT.

Format dasar penulisan alias kolom:

...nama_kolom_asli AS nama_kolom_alias...

nama_kolom_asli adalah nama kolom sesungguhnya yang kita gunakan pada saat pembuatan tabel.

Contohnya adalah nama_dokter dan no_hp.

Nama_kolom_alias adalah alias atau nama lain kolom yang ingin kita gunakan, misalnya kolom no_hp kita aliaskan menjadi nomor_handphone

```
MariaDB [rumah_sakit]> select nama_dokter as nama, no_telp as no_handphone from dokter;
+-----+-----+
| nama | no_handphone |
+-----+-----+
| dr. Marini, Sp.A | 089090909090 |
| dr. Eddy, Sp.P | 087979797979 |
| dr. Santoso, Sp.P | 081230000012 |
| dr. Putri, Sp.M | 084545454545 |
| dr. Anna, Sp.S | 087979797979 |
+-----+
5 rows in set (0.00 sec)
```

Tugas (Take Home) !

1. Buat tabel pegawai sebagai berikut:

Field	Type	Null
Idpegawai	Char(6)	No
Namadepan	Varchar(20)	Yes
namabelakang	Varchar(25)	No
Email	Varchar(20)	No
Telepon	Varchar(20)	Yes
Tglkontrak	date	No
Idjob	Varchar(10)	no
Gaji	Int(8)	Yes
Tunjangan	Int(8)	yes
Idmanajer	Char(6)	yes
iddepartemen	Char(4)	yes

2. Isi data tabel

id pegawai	nama depan	nama belakang	email	telepon	tgl kontrak
E001	Adil	Setiawan	adil@gmail.com	811111213	01/09/2005
E002	Aris	Putra	aris@gmail.com	823113456	01/09/2006
E003	Fais	Sandi	faiz@yahoo.com	856232487	01/10/2006
E004	Emma	Thomson	emma@gmail.com	812229390	01/09/2007
E005	Mike	Tyson	mike@yahoo.com	813457879	01/10/2007

Id job	Gaji	tunjangan	id manajer	Id departemen
L0001	2000000	500000	AL	Coml
L0002	2000000	200000	LE	Coml
L0003	1500000	0	BX	Coml
L0004	1500000	0	CX	Coml
L0005	1500000	0	DX	Coml

3. Tampilkan semua kolom di tabel!
4. Tampilkan kolom idpegawai, namabelakang dan gaji saja!
5. Tampilkan kolom idpegawai, namabelakang, gaji, tunjangan dan sebuah kolom baru yaitu tunjangan+gaji yang berisi jumlah tunjangan dan gaji !
6. Tampilkan kolom idpegawai, namabelakang, gaji, tunjangan dan sebuah kolom baru (gunakan alias) yaitu total_pendapatan yang berisi jumlah tunjangan dan gaji!
7. Tambahkan record baru dengan value: E006,lincoln, burrows, linc@yahoo.com, 085275384544, 2008-09-01, L0006, 1750000, NULL, ex, coml.
8. Untuk pegawai yang ber-id E004 dan E005 ubah idmanajernya menjadi al!
9. Sekarang tampilkan kolom idmanajer saja!
10. Dari percobaan 9, terdapat 3 idmanajer yang sama dengan total record 6, sekarang tampilkan
11. idmanajer tanpa duplikasi idmanajer sehingga akan tampil 4 record dengan idmanajer yang berbeda!
12. Tampilkan pegawai yang gajinya antara 1750000 - 1250000!
13. Tampilkan tabel pegawai yang terurut berdasarkan nama belakang (dari a ke z)!
14. Tampilkan tabel pegawai yang diurutkan berdasarkan nama depan (dari z ke a)!

BAHASA QUERY FORMAL NON PROSEDURAL

BAHASA QUERY FORMAL NON PROSEDURAL

Definisi: pemakai menspesifikasikan data apa yang dibutuhkan tanpa menspesifikasikan bagaimana untuk mendapatkannya.

1. Join Query

Menggabungkan Tabel MySQL dengan INNER JOIN

Tujuan dari menggabungkan tabel adalah untuk menyajikan informasi secara lebih detail. Contohnya dari tabel **dokter** dan tabel **periksa_pasien**, tujuan kita adalah menyajikan informasi data periksa pasien sekaligus nama dokter yang memeriksa pasien tersebut. Kita ingin menyajikan informasi yang berisi nama **kd_periksa**, **diagnosa**, **tindakan** dan **nama dokter yang memeriksa**.

Sintaks:

```
SELECT  nama_kolom_tampil  FROM  nama_tabel1  INNER  JOIN  nama_tabel2  ON  
nama_tabel1.nama_kolom_join_tabel1 = nama_tabel2.nama_kolom_join_tabel2;
```

- **nama_kolom_tampil** adalah nama dari kolom yang akan kita tampilkan, bisa semua kolom dalam tabel, atau hanya kolom tertentu saja.
- **nama_tabel_pertama** adalah nama tabel pertama yang akan digabung.
- **nama_tabel_kedua** adalah nama tabel kedua yang akan digabung.
- **nama_kolom_join_tabel_pertama** adalah nama kolom yang akan digunakan sebagai join dari tabel pertama.
- **nama_kolom_join_tabel_kedua** adalah nama kolom yang akan digunakan sebagai join dari tabel kedua.

Contoh query **SELECT..INNER JOIN..ON** menjadi:

```
MariaDB [rumah_sakit]> select kd_periksa, diagnosa, tindakan, nama_dokter from periksa_pasien  
-> inner join dokter on periksa_pasien.kd_dokter=dokter.kd_dokter;  
+-----+-----+-----+-----+  
| kd_periksa | diagnosa | tindakan | nama_dokter |  
+-----+-----+-----+-----+  
| PP10001 | demam | ambil darah | dr. Marini, Sp.A |  
| PP10002 | bronkitis | rontgen | dr. Eddy, Sp.P |  
| PP10003 | tbc | rontgen | dr. Santoso, Sp.P |  
| PP10004 | minus | laser | dr. Putri, Sp.M |  
+-----+-----+-----+-----+  
4 rows in set (0.00 sec)
```

Menggabungkan Tabel MySQL dengan SELECT..INNER JOIN..USING

Cara **JOIN** kedua adalah menggunakan **USING** sebagai pengganti **ON** untuk query **INNER JOIN**.

Format dasar dari penulisan query **SELECT..INNER JOIN..USING** adalah:

```
SELECT nama_kolom_tampil FROM nama_tabel1 INNER JOIN nama_tabel2 USING (nama_kolom_join);
```

- *nama_kolom_tampil* adalah nama dari kolom yang akan kita tampilkan, bisa semua kolom dalam tabel, atau hanya kolom tertentu saja.
- *nama_tabel_pertama* adalah nama tabel pertama yang akan digabung.
- *nama_tabel_kedua* adalah nama tabel kedua yang akan digabung.
- *nama_kolom_join* adalah nama kolom yang akan digunakan sebagai join.

Syarat untuk **INNER JOIN..USING** adalah ***kedua tabel harus memiliki nama kolom yang sama***. Dalam contoh kita, kolom tersebut adalah kolom *kd_dokter*. Sehingga querynya:

```
MariaDB [rumah_sakit]> select kd_periksa, diagnosa, tindakan, nama_dokter from periksa_pasien
-> inner join dokter using (kd_dokter);
+-----+-----+-----+-----+
| kd_periksa | diagnosa | tindakan | nama_dokter |
+-----+-----+-----+-----+
| PP10001 | demam | ambil darah | dr. Marini, Sp.A |
| PP10002 | bronkitis | rontgen | dr. Eddy, Sp.P |
| PP10003 | tbc | rontgen | dr. Santoso, Sp.P |
| PP10004 | minus | laser | dr. Putri, Sp.M |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Hasilnya sama persis dengan **SELECT..INNER JOIN..ON**, hanya berbeda cara penulisan.

2. Query Multi-tabel

RELASI ANTAR TABEL

Syarat dari query multi tabel adalah adanya relasi antar tabel yang terlibat. Relasi mengisyaratkan adanya persamaan dalam pemakaian sebuah field/kolom. Adapun syarat dalam pembangunan relasi adalah menghubungkan field-field yang sama, penulisannya sama, tipe dan ukurannya sama dan isi yang sama. Relasi ini diketikkan pada klausus *where* query antar tabel untuk tabel-tabel yang digunakan saja.

Sintaks:

```
select nama_tabel1.nama_kolom1, nama_tabel2.nama_kolom2,nama_tabel2.kolom1from
nama_tabel1, nama_tabel2where nama_tabel1.kolom_relati = nama_tabel2.kolom_relati;
```

```
MariaDB [rumah_sakit]> select pasien.nama, dokter.nama_dokter, periksa_pasien.diagnosa, tindakan, biaya_tindakan
-> from pasien, dokter, periksa_pasien
-> where pasien.kd_pasien = periksa_pasien.kd_pasien and
-> dokter.kd_dokter = periksa_pasien.kd_dokter;
+-----+-----+-----+-----+-----+
| nama | nama_dokter | diagnosa | tindakan | biaya_tindakan |
+-----+-----+-----+-----+-----+
| ardi | dr. Marini, Sp.A | demam | ambil darah | 150000 |
| jeffrey | dr. Eddy, Sp.P | bronkitis | rontgen | 300000 |
| asri | dr. Santoso, Sp.P | tbc | rontgen | 300000 |
| firna | dr. Putri, Sp.M | minus | laser | 10000000 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Menggunakan kondisi (where)

Operator *where* berfungsi melakukan spesifikasi query untuk mendapatkan informasi yang akan ditampilkan.

```
MariaDB [rumah_sakit]> select pasien.nama, dokter.nama_dokter, periksa_pasien.diagnosa, tindakan, biaya_tindakan
-> from pasien, dokter, periksa_pasien
-> where pasien.kd_pasien = periksa_pasien.kd_pasien and
-> dokter.kd_dokter = periksa_pasien.kd_dokter
-> and biaya_tindakan >='300000';
+-----+-----+-----+-----+-----+
| nama | nama_dokter | diagnosa | tindakan | biaya_tindakan |
+-----+-----+-----+-----+-----+
| jeffrey | dr. Eddy, Sp.P | bronkitis | rontgen | 300000 |
| asri | dr. Santoso, Sp.P | tbc | rontgen | 300000 |
| firna | dr. Putri, Sp.M | minus | laser | 10000000 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Menggunakan alias (as)

Operator as untuk menggantikan nama tabel untuk meningkatkan efisiensi query.

```
MariaDB [rumah_sakit]> select P.nama, D.nama_dokter, PP.tindakan, biaya_tindakan
-> from pasien P, dokter D, periksa_pasien PP
-> where P.kd_pasien = PP.kd_pasien and D.kd_dokter = PP.kd_dokter
-> and biaya_tindakan >='300000';
+-----+-----+-----+-----+
| nama | nama_dokter | tindakan | biaya_tindakan |
+-----+-----+-----+-----+
| jeffrey | dr. Eddy, Sp.P | rontgen | 300000 |
| asri | dr. Santoso, Sp.P | rontgen | 300000 |
| firna | dr. Putri, Sp.M | laser | 10000000 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

OPERATOR NATURAL JOIN

Operator Natural Join akan melakukan operasi dengan memperlakukan nama-nama kolom yang sama sebagai penghubung antar tabel

```
MariaDB [rumah_sakit]> select pasien.nama, periksa_pasien.tindakan, biaya_tindakan
-> from pasien natural join periksa_pasien;
+-----+-----+-----+
| nama | tindakan | biaya_tindakan |
+-----+-----+-----+
| ardi | ambil darah | 150000 |
| jeffrey | rontgen | 300000 |
| asri | rontgen | 300000 |
| firna | laser | 10000000 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

3. Outer Join

Dengan outer join, tabel akan digabungkan satu arah, sehingga memungkinkan ada data yang NULL (kosong) di satu sisi. Sebagai contoh, kita akan menggabungkan tabel pelanggan dan pesan dimana kita akan menampilkan daftar pelanggan yang pernah melakukan pemesanan (transaksi). Outer Join terbagi menjadi 2 (dua) yaitu LEFT JOIN dan RIGHT JOIN. Berikut ini bentuk umum dan contohnya:

A. LEFT JOIN

Left Join merupakan penggabungan tabel dimana data akan ditampilkan secara keseluruhan pada tabel pertama (kiri) namun record pada tabel kedua (kanan) yang kosong akan ditampilkan dengan isi NULL.

Sintaks:

```
SELECT tabel1.*, tabel2.* FROM tabel1 LEFT JOIN tabel2 ON tabel1.PK=tabel2.FK;
```

```
MariaDB [rumah_sakit]> select pasien.kd_pasien, nama, periksa_pasien.kd_pasien, diagnosa, tindakan
-> from pasien left join periksa_pasien
-> on pasien.kd_pasien = periksa_pasien.kd_pasien;
+-----+-----+-----+-----+-----+
| kd_pasien | nama | kd_pasien | diagnosa | tindakan |
+-----+-----+-----+-----+-----+
| 10001 | ardi | 10001 | demam | ambil darah |
| 10003 | jeffrey | 10003 | bronkitis | rontgen |
| 10005 | asri | 10005 | tbc | rontgen |
| 10002 | firna | 10002 | minus | laser |
| 10004 | glenn | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Berbeda dengan hasil sebelumnya (inner join), penggunaan left join akan menampilkan juga data pasien dengan kd_pasien, walaupun pasien tersebut belum pernah diperiksa. Dan pada kolom kd_pasien, diagnosa dan tindakan untuk pasien 10004 isinya NULL, artinya di tabel kanan (periksa_pasien) pasien belum diperiksa.

B. RIGHT JOIN

Right Join memiliki fungsi yang bertolak belakang dengan left join, dimana right join akan menampilkan data secara keseluruhan pada tabel kedua (kanan), namun NULL pada tabel pertama (kiri).

Sintaks:

```
SELECT tabel1.*, tabel2.* FROM tabel1 RIGHT JOIN tabel2 ON tabel1.PK=tabel2.FK;
```

```
MariaDB [rumah_sakit]> select pasien.kd_pasien, nama, periksa_pasien.kd_pasien, diagnosa, tindakan
-> from pasien right join periksa_pasien
-> on pasien.kd_pasien=periksa_pasien.kd_pasien;
+-----+-----+-----+-----+-----+
| kd_pasien | nama | kd_pasien | diagnosa | tindakan |
+-----+-----+-----+-----+-----+
| 10001 | ardi | 10001 | demam | ambil darah |
| 10003 | jeffrey | 10003 | bronkitis | rontgen |
| 10005 | asri | 10005 | tbc | rontgen |
| 10002 | firna | 10002 | minus | laser |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Hasil perintah right join, tabel yang menjadi acuan adalah tabel sebelah kanan (tabel periksa_pasien), jadi semua isi tabel periksa_pasien akan ditampilkan. Tabel yang ditampilkan akan disesuaikan dengan tabel yang berada disebelah kanan (tabel periksa_pasien).

Tugas

Database : KULIAH

Tabel : MAHASISWA, DOSEN

Tabel MAHASISWA

Field	Tipe data
Nim	Char (12)(PK)
Nama	Varchar (20)
Alamat	Varchar (30)
IPK	Float (10,2)
Id_dosen	Char (5)

Tabel Dosen

Field	Tipe data
Id_dosen	Char (5)(PK)
Nama	Varchar (20)
Alamat	Varchar (30)
Jabatan	Varchar(15)
Notelp	VarChar (15)

Tabel Mahasiswa

NIM	Nama	Alamat	IPK	Id_Dosen
123070201	BUDI SISWANTO	JL MAWAR NO 11	3.01	12346
123070202	SANTI	JL MELATI NO 10	2.75	12344
123070203	INDRA KUSUMA	JL DEMANGAN NO 4	2.83	12345
123070204	KARMAN MAULANA	JL BABARSARI NO 23	2.91	12343
123070205	RIZAD RAHMAN	JL KAPAS NO 8	2.5	12344
123070206	WAWAN ADI PUTRA	JL KLEDOKAN NO 2	3.21	12341
123070207	M TAUFIK HIDAYAT	JL TAMBAKBAYAN NO 3	3.11	12341
123070208	FITRIADI BUDIMAN	JL MERPATI NO 24	3.41	12344
123070209	IDA KUSUMAWATI	JL BANTUL NO 15	3.32	12343
123070210	HIDAYAT NUGRAHA	JL PASIFIK NO 6	2.85	12346

Tabel Dosen

ID_DOSEN	NAMA	ALAMAT	JABATAN	NOTLP
12341	ANDI	JL ELANG NO 11	LEKTOR	8123456789
12342	BUDI	JL AFFANDI NO 17	ASISTEN AHLI	8123456788
12343	SUSI	JL RINGROAD NO 89	LEKTOR	8123456787
12344	SANTI	JL ADI SUCIPTO NO 8	LEKTOR	8123456786
12345	HARI	JL MERAPI NO 23	ASISTEN AHLI	8123456785

Tabel Dosen

- Tampilkan isi tabel DOSEN melalui tabel MAHASISWA
- Tampilkan isi tabel DOSEN dan MAHASISWA yang memiliki ID_DOSEN yang sama.
- Tampilkan isi tabel DOSEN dan MAHASISWA yang memiliki ID_DOSEN yang sama = '12344'.

INNER JOIN

Tabel pelanggan (hanya ditampilkan field id_pelanggan, nm_pelanggan dan email)

id_pelanggan	nm_pelanggan	email
P0001	Achmad Solichin	achmatim@gmail.com
P0002	Budianto	budi@luhur.com
P0003	Hasan	hasan02@yahoo.com
P0004	Amin Riyadi	aminudin@plasa.com

Tabel pesan.

id_pesan	id_pelanggan	tgl_pesan
1	P0001	2008-02-02
2	P0002	2008-02-05
3	P0002	2008-02-10
4	P0004	2008-01-20
5	P0001	2007-12-14

Cara #1. Inner Join dengan WHERE.

Penggabungan dengan klausula WHERE memiliki bentuk umum sebagai berikut:

```
SELECT tabel1.*, tabel2.*  
FROM tabel1, tabel2  
WHERE tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan,  
pesan.tgl_pesan  
FROM pelanggan, pesan  
WHERE pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya sebagai berikut:

id_pelanggan	nm_pelanggan	id_pesan	tgl_pesan
P0001	Achmad Solichin	1	2008-02-02
P0001	Achmad Solichin	5	2007-12-14
P0002	Budianto	2	2008-02-05
P0002	Budianto	3	2008-02-10
P0004	Amin Riyadi	4	2008-01-20

Hasil Penggabungan 2 Tabel dengan WHERE Pada hasil perintah query di atas terlihat bahwa terdapat 5 (lima) transaksi yang dilakukan oleh 3 (tiga) orang pelanggan. Jika kita lihat kembali isi tabel pelanggan di atas, maka terdapat satu pelanggan yang tidak ditampilkan yaitu yang memiliki id pelanggan P0003. Pelanggan tersebut tidak ditampilkan karena belum pernah melakukan transaksi.

Cara #2. Inner Join dengan klausula INNER JOIN.

Berikut ini bentuk umumnya:

```
SELECT tabel1.*, tabel2.*  
FROM tabel1  
INNER JOIN tabel2  
ON tabel1.PK=tabel2.FK;
```

Dan berikut ini perintah SQL penggabungan tabel pelanggan dan pesan.

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan,  
pesan.tgl_pesan  
FROM pelanggan  
INNER JOIN pesan  
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya akan sama dengan gambar di atas (*cara #1*).

Outer Join

LEFT JOIN.

Bentuk umum:

```
SELECT tabel1.*, tabel2.*  
FROM tabel1  
LEFT JOIN tabel2  
ON tabel1.PK=tabel2.FK;
```

Contoh perintah SQL:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan,  
pesan.tgl_pesan  
FROM pelanggan  
LEFT JOIN pesan  
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya:

id_pelanggan	nm_pelanggan	id_pesan	tgl_pesan
P0001	Achmad Solichin	1	2008-02-02
P0001	Achmad Solichin	5	2007-12-14
P0002	Budianto	2	2008-02-05
P0002	Budianto	3	2008-02-10
P0003	Hasan	NULL	NULL
P0004	Amin Riyadi	4	2008-01-20

Hasil Perintah Left Join Berbeda dengan hasil sebelumnya (inner join), penggunaan left join akan menampilkan juga data pelanggan dengan id P0003, walaupun pelanggan tersebut belum pernah bertransaksi. Dan pada kolom id_pesan dan tgl_pesan untuk pelanggan P0003 isinya NULL, artinya di tabel kanan (pesan) pelanggan tersebut tidak ada.

RIGHT JOIN

Bentuk umum:

```
SELECT tabel1.*, tabel2.*  
FROM tabel1  
RIGHT JOIN tabel2  
ON tabel1.PK=tabel2.FK;
```

Contoh perintah SQL:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan, pesan.id_pesan,  
pesan.tgl_pesan  
FROM pelanggan  
RIGHT JOIN pesan  
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya:

id_pelanggan	nm_pelanggan	id_pesan	tgl_pesan
P0001	Achmad Solichin	1	2008-02-02
P0002	Budianto	2	2008-02-05
P0002	Budianto	3	2008-02-10
P0004	Amin Riyadi	4	2008-01-20
P0001	Achmad Solichin	5	2007-12-14

Hasil Perintah Right Join Dengan right join, tabel yang menjadi acuan adalah tabel sebelah kanan (tabel pesan), jadi semua isi tabel pesan akan ditampilkan. Jika data pelanggan tidak ada di tabel pelanggan, maka isi tabel pesan tetap ditampilkan.

Tugas

Dengan menggunakan data yang sama pada sebelumnya(Tabel Mahasiswa dan Dosen) :

- Tampilkan isi tabel DOSEN melalui tabel MAHASISWA. (Inner Join)
- Tampilkan isi tabel DOSEN dan MAHASISWA dengan LEFT JOIN.
- Tampilkan isi tabel DOSEN dan MAHASISWA dengan RIGHT JOIN.

BAHASA QUERY FORMAL KOMERSIAL

1. Subquery

Sub Query adalah query didalam query. Artinya seleksi data berdasarkan dari hasil seleksi data yang telah ada. Sintak SQL nya sama dengan sintak SQL pada umumnya, hanya saja kondisi setelah where atau from diikuti dengan query baru atau sub query.

Sintaks:

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

```
MariaDB [rumah_sakit]> select pasien.nama, periksa_pasien.biaya_tindakan
-> from pasien, periksa_pasien
-> where pasien.kd_pasien=periksa_pasien.kd_pasien
-> and periksa_pasien.biaya_tindakan = (select max(biaya_tindakan)from periksa_pasien);
+-----+
| nama | biaya_tindakan |
+-----+
| firna |      10000000 |
+-----+
1 row in set (0.00 sec)
```

Keuntungan utama dari subquery adalah:

- Subquery membuat query-query menjadi tersusun, sehingga ada kemungkinan untuk memisahkan / memberi batasan area untuk setiap bagian statement.
- Subquery menyediakan cara alternatif untuk menjalankan operasi-operasi yang dalam keadaan lain akan membutuhkan JOIN dan UNION yang kompleks.
- Subquery lebih mudah dibaca dibanding JOIN atau UNION yang kompleks

2. Exist dan not exist

A. Exist

Merupakan jenis operator boolean, yang menghasilkan nilai benar (true) atau salah (false). Operator exist akan memberikan nilai benar (true) kalau sub query menghasilkan paling tidak sebuah baris/record.

Sintaks:

```
SELECT * FROM t1 WHERE exist = (SELECT column1 FROM t2);
```

Contoh: menampilkan nama pasien yang sudah memiliki kode periksa

```
MariaDB [rumah_sakit]> select nama from pasien
-> where exists (select * from periksa_pasien
-> where kd_pasien=pasien.kd_pasien);
+-----+
| nama |
+-----+
| ardi |
| firna |
| jeffrey |
| asri |
+-----+
4 rows in set (0.00 sec)
```

B. Not Exists

Menampilkan nama pasien yang belum memiliki kode periksa

```
MariaDB [rumah_sakit]> select nama from pasien
-> where not exists (select * from periksa_pasien
-> where kd_pasien=pasien.kd_pasien);
+-----+
| nama |
+-----+
| glenn |
+-----+
1 row in set (0.00 sec)
```

3. Any dan All

A. Any

Operator any hampir sama dengan Exist. Tetapi operator relasi yang digunakan biasanya selain = (sama dengan). Hal tersebut apabila menggunakan operator relasi = maka fungsinya akan sama dengan operator IN.

```
MariaDB [rumah_sakit]> select kd_periksa, biaya_tindakan
-> from periksa_pasien
-> where biaya_tindakan > any (select biaya_tindakan from periksa_pasien);
+-----+
| kd_periksa | biaya_tindakan |
+-----+
| PP10002    |      300000 |
| PP10003    |      300000 |
| PP10004    |  10000000 |
+-----+
3 rows in set (0.00 sec)
```

Hasil query akan menampilkan kode periksa dan biaya tindakan selain yang tidak paling rendah.

B. All

Operator all digunakan untuk melakukan pembandingan dengan sub query. Kondisi dengan all menghasilkan nilai benar jika pembandingan menghasilkan benar untuk setiap nilai dalam sub query.

```
MariaDB [rumah_sakit]> select biaya_tindakan from periksa_pasien
-> where biaya_tindakan < all (select biaya_tindakan from periksa_pasien
-> where kd_periksa = 'PP10003');
+-----+
| biaya_tindakan |
+-----+
|      150000 |
+-----+
1 row in set (0.00 sec)
```

Hasil query menampilkan biaya tindakan yang memiliki harga di bawah kode periksa 'PP10003'.

Tugas

1. Buatlah database Perkuliahannya yang terdiri dari tabel sebagai berikut:

Tabel mahasiswa

Nim	Nama	jk	agama	alamat	kelas
0802100011	emy	Perempuan	Islam	Jakarta	A11
0802100012	waldan	Laki-laki	Islam	Bekasi	B13
0802100013	tasya	Perempuan	Protestan	Karawang	B13
0802100014	rika	Perempuan	Katolik	Cikampek	A13
0802100015	thomas	Laki-laki	Katolik	Cikarang	A12

Tabel matakuliah

kodemk	namamk	sks
DKT01	Pemrogram Komputer	3
KKT05	Basis Data	4
KKT07	Sistem Informasi Manajemen	3
PKT21	Bahasa Inggris	2
DKT33	Algoritma Pemrograman	3

Tabel khs

nim	kodemk	semester	TA	nilai
0802100011	DKT01	3	2016/2017	A
0802100012	PKT21	5	2016/2017	A
0802100013	PKT21	5	2016/2017	B
0802100014	DKT01	2	2016/2017	C
0802100015	KKT07	4	2016/2017	B

Tabel Dosen

nip	namadosen	jk	alamat	telp	pendidikan
132312490	Susi	Perempuan	Bekasi	08122771344	S2
132312478	Putri	Perempuan	Karawang	085678900123	S3
132312455	Ratna	Perempuan	Jakarta	08123456779	S2
132312445	Mirza	Laki-laki	Cibitung	08113456888	S3
132312433	Adil	Laki-laki	Tambun	08117890123	S2

Tabel Jurusan

kodejur	namajur	ketua
S1TI	Teknik Informatika S1	132312490
S1SI	Sistem Informasi S1	132312478
D3MI	Manajemen Informatika D3	132312455
D3KA	Komputerisasi Akuntansi D3	132312445
D3TK	Teknik Komputer D3	132312433

Tabel Kelas

kelas	kodejur	nip	dosenwali
S1TI3A	S1TI	132312490	132312490
S1TI3C	S1TI	132312490	132312490
D3MI1B	D3MI	132312455	132312455
S1SI5A	S1SI	132312478	132312478

1. Menampilkan daftar mahasiswa yang terdapat dalam tabel mahasiswa dan tabel khs.

mysql>select nim,nama

from mahasiswa

where exists (select * from khs where nim=mahasiswa.nim);

2. Tulis perintah :

mysql>select nim,nama

from mahasiswa

where nim = any (select distinct nim from khs);

3. Tulis perintah :

mysql>select nim,nama

from mahasiswa

where nim in (select distinct nim from khs);

4. Tulis perintah :

mysql>select nim,nama

from mahasiswa

where not exists (select * from khs where nim=mahasiswa.nim);

- *Gunakan tabel mahasiswa, kelas, jurusan dan dosen untuk menampilkan data berikut:*
 - a. menampilkan data dosen (nip, nama) yang menjabat sebagai ketua jurusan sekaligus menjadi dosen wali.
 - b. Menampilkan data dosen (nip,nama) yang tidak menjabat sebagai ketua jurusan.
 - c. Menampilkan data dosen (nip, nama) yang tidak menjadi dosen wali.
 - d. Menampilkan daftar mahasiswa yang dibimbing oleh seorang dosen wali.
 - e. Menampilkan daftar mahasiswa suatu jurusan.
 - f. Menampilkan daftar mahasiswa suatu jenjang program studi.

Fungsi Agregat

❖ Fungsi Agregat

Fungsi agregat dalam MySQL adalah fungsi yang menerima koleksi nilai dan mengembalikan nilai tunggal sebagai hasilnya, seperti: jumlah data, nilai minimum, nilai maximum dan nilai rata-rata .

a. SUM

Untuk menghitung total nilai dari kolom tertentu

Sintaks: Select SUM (nama_field_nilai) from nama_tabel;

b. COUNT

Untuk menghitung jumlah record.

Sintaks: Select COUNT (*) from nama_tabel;

c. AVG

Untuk menampilkan nilai rata-rata dari suatu kolom.

Sintaks: Select nama_kolom, AVG (nama_tabel) from nama_tabel;

```
MariaDB [mahasiswa]> select nama_matkul,avg(nilai) as rata_rata
    -> from nilai
    -> where nama_matkul='Algoritma';
+-----+-----+
| nama_matkul | rata_rata |
+-----+-----+
| Algoritma   |      89 |
+-----+-----+
1 row in set (0.00 sec)
```

d. MAX

Untuk menampilkan nilai tertinggi dari suatu kolom.

Sintaks: Select nama_kolom, MAX (nama_tabel) from nama_tabel;

```
MariaDB [mahasiswa]> select nama_mhs, nama_matkul, max(nilai) as nilai_tertinggi
    -> from nilai
    -> where nama_matkul='Algoritma';
+-----+-----+-----+
| nama_mhs | nama_matkul | nilai_tertinggi |
+-----+-----+-----+
| Paul     | Algoritma   |      89 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

e. MIN

Untuk menampilkan nilai terendah dari suatu kolom.

Sintaks: Select nama_kolom, MIN (nama_tabel) from nama_tabel;

```

MariaDB [mahasiswa]> select nama_mhs, nama_matkul, min(nilai) as nilai_terendah
-> from nilai
-> where nama_matkul='PTI';
+-----+-----+-----+
| nama_mhs | nama_matkul | nilai_terendah |
+-----+-----+-----+
| Adi      | PTI        | 70          |
+-----+-----+-----+
1 row in set (0.00 sec)

```

❖ Group By

Tampilkan nilai terbesar dan terkecil untuk setiap mahasiswa

Query untuk menyelesaikan kasus ketiga di atas sebenarnya sama saja dengan yang sebelumnya. Perbedaannya hanya pada fungsi agregat yang digunakan untuk menampilkan nilai terbesar dan terkecil yaitu MAX() dan MIN(). Berikut ini query dan hasil query-nya.

```
SELECT nim, nama, MAX(nilai) as terbesar, MIN(nilai) as terkecil FROM nilai GROUP BY nim;
```

```

MariaDB [mahasiswa]> select nim, nama_mhs, max(nilai) as terbesar, min(nilai) as terkecil
-> from nilai group by nim;
+-----+-----+-----+-----+
| nim      | nama_mhs | terbesar | terkecil |
+-----+-----+-----+-----+
| 2013100090 | Paul     | 89       | 89       |
| 2013100111 | Adi      | 90       | 65       |
| 2013100391 | Jessica  | 89       | 69       |
| 2013100598 | Adi      | 88       | 70       |
| 2013100760 | Sylvia   | 89       | 65       |
| 2013100777 | Shinta   | 95       | 68       |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

Tampilkan rata-rata nilai yang didapat mahasiswa untuk setiap matakuliah

Cukup jelas bahwa pada kasus ini, mirip dengan kasus kedua di atas, namun pengelompokan data berdasarkan matakuliah, bukan berdasarkan mahasiswa. Querynya kurang lebih sebagai berikut:

```
SELECT matkul, AVG(nilai) as rata_rata FROM nilai GROUP BY matkul;
```

```

MariaDB [mahasiswa]> select nama_matkul, avg(nilai) as rata_rata
-> from nilai group by nama_matkul;
+-----+-----+
| nama_matkul | rata_rata |
+-----+-----+
| Algoritma    | 89        |
| Algotirma   | 79.6      |
| Anapersi    | 84.6      |
| Anapersi Lanjutan | 83.8      |
| Biomedis    | 83.8      |
| Cloud Computing | 75        |
| Mobile Computing | 79.4      |
| PBO          | 77.8      |
| Pemograman Web | 83.6      |
| PTI          | 81.6      |
| Statistik    | 78.4      |
+-----+-----+
11 rows in set (0.00 sec)

```

❖ Having

Tampilkan rata-rata nilai untuk setiap mahasiswa, yang rata-rata nilai lebih besar dari 80

```

MariaDB [mahasiswa]> select nim, nama_mhs, avg(nilai) as rata_rata
-> from nilai group by nim
-> having avg(nilai)>80;
+-----+-----+-----+
| nim   | nama_mhs | rata_rata |
+-----+-----+-----+
| 2013100090 | Paul    |      89 |
| 2013100111 | Adi     |     81.3 |
| 2013100760 | Sylvia  |     81.1 |
| 2013100777 | Shinta  |      83 |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

❖ Operasi Himpunan

- UNION

Berguna untuk menggabungkan beberapa query SELECT untuk menghasilkan satu keluaran saja.

Batasan/aturan untuk menerapkan union adalah sebagai berikut:

- Query yang disatukan harus menghasilkan jumlah field yang sama.
- Nilai record yang sama dalam UNION akan disatukan, dan tidak akan tampil dua kali (sama ketika kita menggunakan DISTINCT).
- Statement ORDER menggunakan alias pada setiap SELECT, bukan nama field sebenarnya.
- Statement ORDER (tanpa LIMIT pada salah-satu SELECT) harus disimpan di akhir, karena jika disimpan didalam salah-satu SELECT tidak akan berpengaruh.
- Jika menggunakan statement LIMIT (dan atau ORDER), harus ditentukan didalam salah-satu SELECT atau LIMIT total.

Sintaks:

```

Select nama_field1, nama_field2 from nama_tabel1
Union
Select nama_field1, nama_field2 from nama_tabel2
Order by nama_field asc;

```

```

MariaDB [mahasiswa]> select nip, nama_dosen from daftar_dosen
-> union
-> select kd_matkul, nama_matkul from mata_kuliah
-> order by nip asc;
+-----+-----+
| nip   | nama_dosen |
+-----+-----+
| L10001 | Sabrina Sari
| L10002 | Maya Ari Putri
| L10003 | Susi Indriani
| L10004 | Tia Santrini
| L10005 | M. Siddiq
| L10006 | Rubin Hadi
| L10007 | Mustalifah
| L10008 | Arif Budiman
| MK001  | Algoritma
| MK002  | Statistik
| MK003  | PBO
| MK004  | Pemograman Web
| MK005  | PTI
| MK006  | Mobile Computing
| MK007  | Cloud Computing
| MK008  | Anapersi
| MK009  | Anapersi Lanjutan
| MK010  | Biomedis
+-----+-----+
18 rows in set (0.00 sec)

```

- INTERSECT

Operator yang digunakan untuk memperoleh data dari dua buah *query* dimana data yang ditampilkan adalah yang memenuhi kedua *query* tersebut dengan ketentuan jumlah, nama dan tipe kolom dari masing-masing tabel yang akan ditampilkan datanya harus sama.

**Akan tetapi, pada mysql tidak dapat menerapkan intersect dan digantikan dengan sintaks join.

Sintaks:

```

Select nama_tabel1.nama_field1, nama_tabel2.nama_field2
from nama_tabel1
join nama_tabel2
on nama_tabel1.nama_field1= nama_tabel2.nama_field1;

```

```

MariaDB [mahasiswa]> select daftar_dosen.nip, daftar_dosen.nama_dosen, mata_kuliah.nama_matkul
-> from daftar_dosen
-> join mata_kuliah
-> on daftar_dosen.nip=mata_kuliah.nip;
+-----+-----+-----+
| nip   | nama_dosen | nama_matkul |
+-----+-----+-----+
| L10004 | Tia Santrini | Algoritma
| L10001 | Sabrina Sari | Statistik
| L10002 | Maya Ari Putri | PBO
| L10006 | Rubin Hadi | Pemograman Web
| L10005 | M. Siddiq | PTI
| L10008 | Arif Budiman | Mobile Computing
| L10002 | Maya Ari Putri | Cloud Computing
| L10001 | Sabrina Sari | Anapersi
| L10006 | Rubin Hadi | Anapersi Lanjutan
| L10003 | Susi Indriani | Biomedis
+-----+-----+-----+
10 rows in set (0.00 sec)

```

- Minus

Operator minus merupakan kebalikan dari intersect, digunakan untuk memperoleh data dari dua buah query dimana data yang ditampilkan adalah yang tidak memenuhi dari kedua query tersebut dengan ketentuan jumlah, nama dan tipe kolom dari masing-masing tabel yang akan ditampilkan datanya harus sama.

**Akan tetapi, pada mysql tidak dapat menerapkan minus dan digantikan dengan sintaks left join dan right join.

Sintaks:

```
Select nama_tabel1.nama_field1, nama_tabel2.nama_field2  
from nama_tabel1 left join nama_tabel2  
on nama_tabel1.nama_field1= nama_tabel2.nama_field1  
where nama_tabel2.nama_field1 is null;
```

```
MariaDB [mahasiswa]> select dd.nip, dd.nama_dosen, mk.nama_matkul  
-> from daftar_dosen as dd left join mata_kuliah as mk  
-> on dd.nip=mk.nip  
-> where mk.nip is null;  
+-----+-----+-----+  
| nip    | nama_dosen | nama_matkul |  
+-----+-----+-----+  
| L10007 | Mustalifah | NULL       |  
+-----+-----+-----+  
1 row in set (0.00 sec)
```

Tugas

Buatlah table :

Tabel Mahasiswa

	npm	nama	kota	tgl_lahir
	30100231	Andri	Jakarta	6/16/82
	32100100	Feni	Bogor	12/25/83
	32100123	Gerhana	Bogor	5/14/82
	33100065	Vvati	Depok	12/12/82
	33100144	Vera	Bogor	9/8/82
	33100256	Yudha	Jakarta	2/20/81
▶				

Tabel Mata kuliah

	kd_mk	nama_mk	sks
	KD000123	DASAR AKUNTANSI 1	2
	KK000111	PRAK. FISIKA DASAR	1
	KK000231	SISTEM BASIS DATA	2
	KK000244	BAHASA RAKITAN	3
	KU000001	KEWIRAAAN	2
▶			

Tabel Mata kuliah

	npm	kd_mk	nil_mid	nil_uas
	30100231	KK000244	70	50
	32100100	KK000001	80	85
	32100123	KK000231	50	60
	33100144	KD000123	65	99
	33100256	KK000231	60	80
▶			0	0

- Tampilkan semua data pada tabel MHS
- Tampilkan record tertentu pada tabel MTKULIAH
- Dapatkan nama mahasiswa yang mendapatkan nilai uas > 75
- Dapatkan nama mahasiswa yang mengambil mata kuliah basis data
- Ubahlah sks matakuliah basis data menjadi 3 sks
- Dapatkan nama mahasiswa yang tinggal di bogor dan mengambil mata kuliah bahasa rakitan

Daftar Pustaka

1. Fatansyah. 2012. Basis Data. Bandung: Informatika. ISBN:978-602-8758-53-6
2. Ramakrishnan, Raghu & Gehrke, Johanness. 2003. *Database Management Systems*. McGraw Hill Education, New York . ISBN: 979-731-524-X
3. Date, CJ. 2000. *An Introduction to Database System Seventh Edition*. New Jersey: Pearson Addison Weesley. ISBN: 979-683-185-6