

# Pengenalan Mysql

## Pertemuan 1

## Basis Data

### Definisi Basis Data

Sebelum masuk ke materi *basis data* kita harus mengetahui pengertian atau definisi dari *basis data* itu sendiri, Jadi **Basis Data** ialah **Wadah atau tempat berkumpulnya informasi yang terstruktur maupun tidak terstruktur.**

#### *Basis dan data* >

Basis : Wadah atau tempat berkumpul

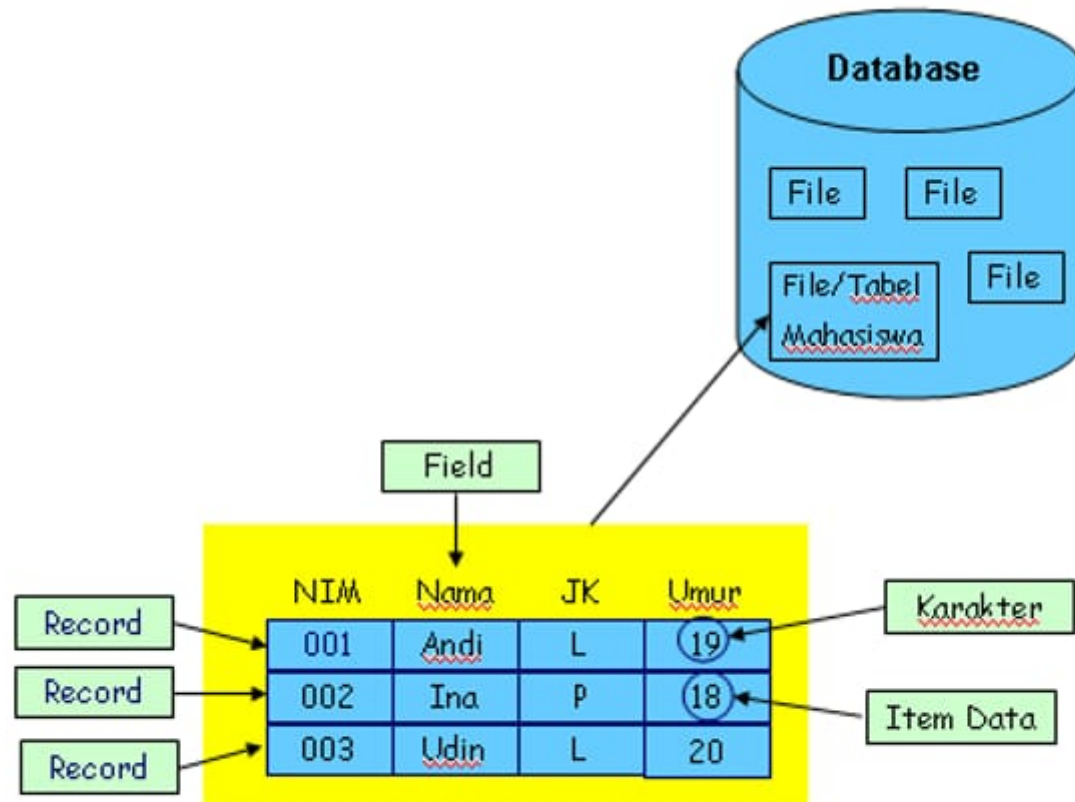
Data : Kumpulan fakta dari sebuah objek

### Peranan Basis Data

Basis data memiliki peranan yang sangat penting dalam pengolahan data agar terstruktur dengan baik, saya ambil contoh dari perusahaan **PDAM**, Terdapat ruangan khusus CPU Server yang berisikan data data dari konsumen termasuk di antaranya nama, tempat tinggal, dan informasi informasi lain yang dibutuhkan Oleh oleh perusahaan.

## Struktur Basis Sata

**Basis data** umumnya digambarkan sebagai tabung yang berisikan data data, yang dimana data tersebut berbentuk tabel, contohnya ialah sebagai berikut :



## Contoh Table

berikut ialah contoh table simpel yang berisikan item Nama,Hoby dan umur :

| Nama   | Hoby  | Umur |
|--------|-------|------|
| Fachri | Makan | 16   |
| Hayril | Minum | 16   |

| Nama   | Hoby    | Umur |
|--------|---------|------|
| Rahmat | Membaca | 17   |
| Rahman | Bermain | 17   |
| Juli   | Tidur   | 16   |

## Penjelasan Database

Database ialah tempat disimpan nya data data yang digambarkan sebagai bentuk tabung, lalu berisikan data atau sebuah file dalam bentuk tabel. lalu dalam tabel dibagi dalam beberapa bagian yaitu :

- **Field** : kumpulan karakter yang membentuk sebuah arti disebut sebagai field.
- **Record** : kumpulan field dalam bentuk yang kompleks
- **item data** : Sebuah nilai data yang berada pada kolom
- **karakter** : sebuah nilai yang terletak pada sebuah kolom by

## Pertemuan 2

## Mysql

**Mysql** ialah software yang digunakan untuk membuat sebuah *database*. dan berikut ialah tahapan cara mendownload serta penggunaan awal dari mysql mulai dari pembuatan database dan pembuatan table.

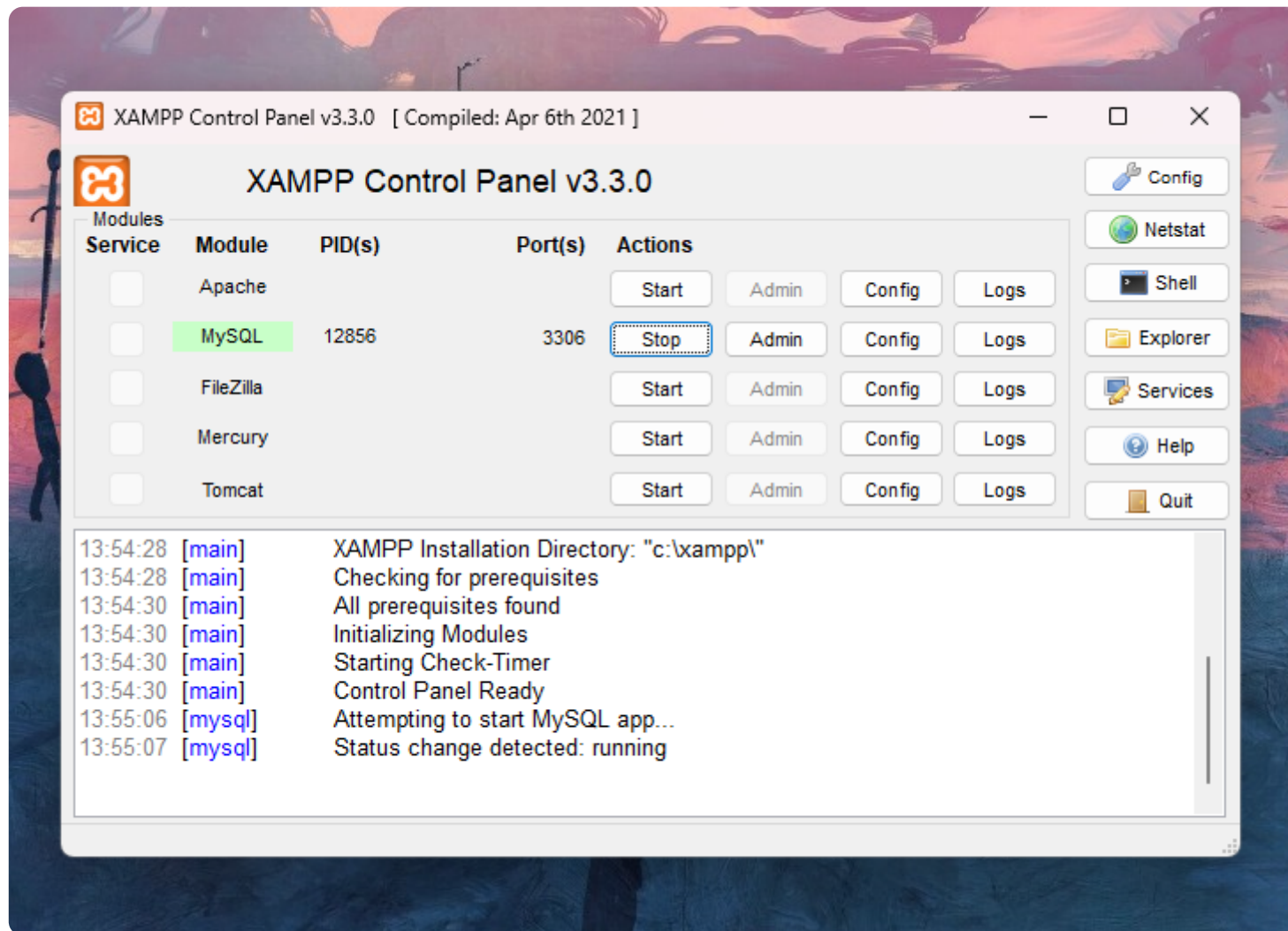
## Cara Penginstalan Mysql

untuk menginstal mysql pertama tama kita harus memingstal **xampp** dan untuk mendapatkannya kita dapat mendownload software tersebut pada browser. dan setelah menginstal ikuti tahap berikut ini :

1. Buka **XAMPP** lalu tunggu proses instaling selesai. dan setelah itu klik icon yang bertuliskan "Flnish"



2. Setelah terlihat tampilan seperti ini klik icon "Start" pada baris **MySQL** dan klik icon yang bertuliskan "Shell".



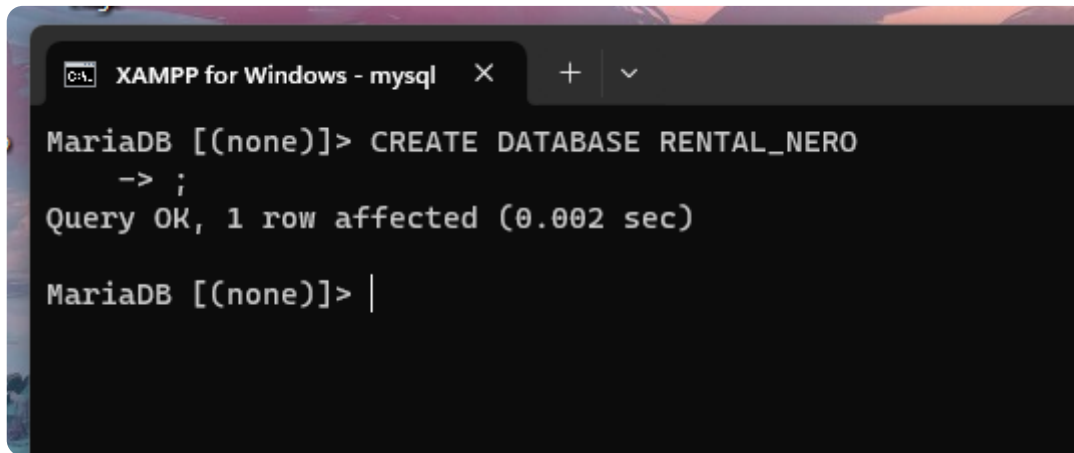
3. Dan setelah itu kita akan langsung di arahkan ke dalam **Mysql** dan tahap penginstalan telah selesai.

## Tentang Database

setelah melakukan tahap "penginstalan Mysql" selanjutnya ada tahap *membuat*, *menghapus*, *melihat* dan *menggunakan* data base.

## Membuat Database

untuk membuat database kita menggunakan query `CREATE DATABASE [nama_database]` setelah mengetikkan query tersebut akan terbuat sebuah database dan hasilnya akan seperti dibawah ini :

A screenshot of a MySQL command prompt window titled "XAMPP for Windows - mysql". The prompt shows the command "CREATE DATABASE RENTAL\_NERO" being entered and executed. The output indicates "Query OK, 1 row affected (0.002 sec)".

```
MariaDB [(none)]> CREATE DATABASE RENTAL_NERO
-> ;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> |
```

## Menampilkan Database

untuk menampilkan *data base* kita bisa menggunakan `SHOW DATABASES;` untuk menampilkan data base yang telah digunakan seperti pada contoh di atas kita telah membuat *database* dengan nama "RENTAL\_NERO" dan hasilnya tampil pada daftar database yang telah kita buat, hasilnya akan terlihat seperti dibawah ini :

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| rental_nero |
| test |
+-----+
6 rows in set (0.002 sec)

MariaDB [(none)]> |
```

## Menghapus Database

adapun untuk menghapus *database* kita menggunakan query `DROP DATABASE [nama_database]` dan *database* yang ditentukan akan terhapus lalu untuk memastikannya kita dapat melihatnya di `SHOW DATABASES` , setelah menuliskan query `DROP` hasilnya akan seperti dibawah ini :

```
MariaDB [(none)]> DROP DATABASE RENTAL_NERO
-> ;
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> |
```

## Menggunakan Database



lalu yang terakhir untuk menggunakan *database* yang telah kita buat di langkah langkah sebelumnya, kita dapat menggunakan query `USE [nama_database]` , lalu setelah mengetikkan query diatas kita akan langsung masuk kedalam *database* yang kita inginkan. Hasilnya akan terlihat seperti dibawah ini :

```
MariaDB [(none)]> USE RENTAL_NERO;  
Database changed  
MariaDB [RENTAL_NERO]> |
```

## Tipe Data Pada Mysql

### Angka

- **INT**: Untuk menyimpan nilai bilangan bulat (integer). Misalnya, INT dapat digunakan untuk menyimpan angka seperti 1, 100, -10, dan sebagainya.
- **DECIMAL**: Digunakan untuk menyimpan nilai desimal presisi tinggi, cocok untuk perhitungan finansial atau keuangan.
- **FLOAT** dan **DOUBLE**: Digunakan untuk menyimpan nilai desimal dengan presisi floating-point. DOUBLE memiliki presisi lebih tinggi dibandingkan FLOAT.
- **TINYINT**, **SMALLINT**, **MEDIUMINT**, dan **BIGINT**: Tipe data ini menyimpan bilangan bulat dengan ukuran yang berbeda-beda.

Contoh :

```
mysql CREATE TABLE contoh_tabel ( id INT, harga DECIMAL(10, 2), jumlah_barang TINYINT );
```

Dalam contoh tersebut, id menggunakan tipe data INT, harga menggunakan tipe data **DECIMAL** dengan presisi 10 digit dan 2 angka di belakang koma, dan jumlah\_barang menggunakan tipe data **TINYINT**.

# Teks

- `==CHAR(N) ==` Menyimpan string karakter tetap dengan panjang N. Contoh: `==CHAR(10) ==` akan menyimpan string dengan panjang tepat 10 karakter.
- `VARCHAR(N)`: Menyimpan string karakter dengan panjang variabel maksimal N. Misalnya, `==VARCHAR(255) ==` dapat menyimpan string hingga 255 karakter, tetapi sebenarnya hanya menyimpan panjang yang diperlukan plus beberapa overhead.
- `==TEXT:` ==Digunakan untuk menyimpan teks dengan panjang variabel, tanpa batasan panjang tertentu. Cocok untuk data teks yang panjangnya tidak terduga.
- `==ENUM:` ==Memungkinkan Anda mendefinisikan set nilai yang mungkin dan membatasi kolom hanya dapat mengambil salah satu dari nilai tersebut.
- `==SET:` ==Mirip dengan ENUM, namun dapat menyimpan satu atau lebih nilai dari himpunan yang telah ditentukan.

**Contoh :**

```
CREATE TABLE contoh_tabel (  
    nama CHAR(50),  
    alamat VARCHAR(100),  
    catatan TEXT,  
    status ENUM('Aktif', 'Non-Aktif')  
);
```

# Tanggal

- `DATE` : Menyimpan nilai tanggal dengan format YYYY-MM-DD.
- `TIME`: Menyimpan nilai waktu dengan format HH:MM:SS.
- `==DATETIME:` ==Menggabungkan nilai tanggal dan waktu dengan format YYYY-MM-DD HH:MM:SS.
- `==TIMESTAMP:` ==Sama seperti DATETIME, tetapi dengan kelebihan diatur secara otomatis saat data dimasukkan atau diubah.

```
CREATE TABLE ContohTabel (  
    tanggal DATE,  
    waktu TIME,  
    datetimekolom DATETIME,  
    timestampkolom TIMESTAMP  
);
```

Dalam contoh ini, kolom **tanggal** akan menyimpan nilai tanggal, **waktu** menyimpan nilai waktu, **\*datetimekolom** menyimpan kombinasi tanggal dan waktu, dan **\*timestampkolom** akan secara otomatis diatur saat data dimasukkan atau diubah.

## Boolean

- **BOOL / BOOLEAN / TINYINT(1)**: Digunakan untuk menyimpan nilai boolean, yang dapat mewakili kebenaran atau kesalahan. Representasi nilai benar adalah 1, sedangkan nilai salah direpresentasikan sebagai 0. Meskipun nilai selain 0 dianggap benar, secara umum, ketiganya seringkali digunakan secara bergantian. Seringkali, ketika Anda mendeklarasikan kolom sebagai BOOL atau BOOLEAN, MySQL mengonversinya secara otomatis menjadi TINYINT(1), yang juga dapat digunakan untuk menyimpan nilai boolean dengan 0 untuk false dan 1 untuk true.

### 1. Menggunakan BOOLEAN

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed BOOLEAN  
);
```

Dalam contoh diatas, kita mendefinisikan kolom completed sebagai tipe data BOOLEAN. Ini merupakan cara yang sah dan umum digunakan di MySQL. Nilai yang dapat disimpan dalam kolom ini adalah TRUE atau FALSE, atau dalam representasi angka, 1 atau 0.

## 2. Menggunakan BOOL

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed BOOL  
);
```

Dalam contoh ini, kita menggunakan BOOL sebagai tipe data untuk kolom completed. Perlu dicatat bahwa MySQL secara otomatis mengonversi BOOL menjadi TINYINT(1). Oleh karena itu, ini setara dengan contoh pertama. Namun, beberapa pengembang lebih suka menggunakan BOOLEAN untuk kejelasan.

## 3. Menggunakan TINYINT(1)

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed TINYINT(1)  
);
```

Dalam contoh ini, kita menggunakan TINYINT(1) sebagai tipe data untuk kolom completed. Ini adalah pendekatan yang valid karena MySQL mengonversi BOOL menjadi TINYINT(1) secara otomatis. Dalam hal ini, nilai yang dapat disimpan adalah 1 untuk TRUE dan 0 untuk FALSE.

# Tentang Table

setelah kita membuat *database* selanjutnya kita membuat *tabel* untuk tempat menyimpan data data yg akan disimpan.

# Pembuatan Tabel

untuk pembuatan table kita menggunakan format seperti di bawah ini :

```
CREATE TABLE [nama_table] (  
    -> id_pelanggan INT (4) PRIMARY KEY NOT NULL,  
    -> nama_depan VARCHAR(20) NOT NULL,  
    -> nama_belakang VARCHAR(20) NOT NULL,  
    -> no_telp VARCHAR (12) UNIQUE  
    );
```

Dan hasilnya akan seperti berikut :

```
MariaDB [RENTAL_NERO]> CREATE TABLE Pelanggan (  
    -> Id_Pelanggan int(4) PRIMARY KEY NOT NULL,  
    -> Nama_Depan VARCHAR(20) NOT NULL,  
    -> Nama_Belakang VARCHAR(20) NOT NULL,  
    -> No_Telp CHAR(12) UNIQUE );  
Query OK, 0 rows affected (0.050 sec)
```

## Analisis >

id\_pelanggan,nama\_depan,nama\_belakang dan no\_telp : Menjadi field dari sebuah table

INT : tipe data untuk menampung nilai angka

VARCHAR : tipe data untuk menampung nilai huruf

( ) : untuk menentukan nilai dari tipe data

PRIMARY KEY : menjadi sebuah penanda karna hanya ada satu di sebuah kolom

NOT NULL : berarti kolom harus di isi

UNIQUE : memiliki sistem yang mirip dengan primary key.

## Kesimpulan :

kata pertama (id\_pelanggan) ialah nama dari kolomnya lalu kata keduanya (INT) ialah nama tipe datanya lalu kata ketiga (NOT NULL) ialah untuk memastikan agar kolomnya harus di isi atau tidak.

## Menampilkan Struktur Tabel

lalu setelah membuat tabel kita dapat menampilkan struktur dari table yang telah kita buat dengan cara mengetik `DESC [nama_tabel]` dan hasil kodenya akan seperti berikut :

```
DESC PELANGGAN;
```

Dan hasilnya akan seperti berikut :

```
MariaDB [RENTAL_NERO]> DESC PELANGGAN;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Id_Pelanggan | int(4)    | NO   | PRI | NULL    |       |
| Nama_Depan   | varchar(20) | NO   |     | NULL    |       |
| Nama_Belakang | varchar(20) | NO   |     | NULL    |       |
| No_Telp     | char(12)  | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.037 sec)
```

### Analisis >

DESC : berfungsi untuk mendeskripsikan hasil atau struktur tabel yang telah dibuat

### Kesimpulan :

setelah membuat table kita dapat menampilkan struktur dari table tersebut dengan mengetikkan query `DESC PELANGGAN` dan hasilnya akan tampil.

## Menampilkan Daftar Tabel

untuk menampilkan daftar tabel yang ada dalam *database* kita menggunakan query `SHOW TABLES;` dan hasil yang akan tampil ialah seperti berikut :

```
MariaDB [RENTAL_NERO]> SHOW TABLES;
+-----+
| Tables_in_rental_nero |
+-----+
| pelanggan             |
+-----+
1 row in set (0.001 sec)
```

### Analisis >

- `SHOW TABLES` ialah query yang berfungsi untuk menampilkan table table yang terdapat pada databse yang kita gunakan.
- `DESC TABLES` ialah query yang berfungsi untuk menampilkan struktur table yang telah dibuat.
- Kita dapat membuat table dalam sebuah database dengan mengetik format pada nomor 1.

### Kesimpulan :

Setelah membuat table, kita dapat menampilkan daftar table table yang telah kita buat dengan query `SHOW TABLES;` dan seluruh table yang telah kita buat akan tampil berurutan seperti pada contoh di bawah.

# Sesi QNA

## Pertanyaan :

1. Mengapa hanya kolom id\_pelanggan yang menggunakan constraint PRIMARY KEY?
2. Mengapa pada kolom no\_telp yang menggunakan data CHAR bukan VARCHAR?
3. Mengapa hanya kolom no\_telp yang menggunakan constraint UNIQUE?
4. Mengapa kolom no\_telp tidak memakai constraint NOT NULL sementara kolom lainnya menggunakan constraint tersebut?
5. Tuliskan perbedaan antara PRIMARY KEY dengan UNIQUE?

## Jawaban :

1. Untuk membedakan id Pelanggan yang sama, mencegah duplikasi, dan mempermudah pencarian data.
2. Tipe data char menyimpan data dalam karakter panjang lebih efisien. pencarian pada kolom tipe data **CHAR** dapat lebih cepat.
3. Karna no\_telp tidak ada yang sama semua pasti berbeda dan nilainya unik maka menggunakan constraints unique artinya data dalam tabel id\_telpon berbeda tidak ada yang sama.
4. Nomor telpon dianggap opsional. nomor telepon hanya menjadi wajib saat pengguna melakukan langkah-langkah tertentu, Anda mungkin tidak ingin mengharuskan pengguna mengisinya pada tahap awal.
5. PRIMERY KEY untuk membedakan data yang sama dan hanya boleh 1 dan tidak boleh tidak ada.  
Kalau UNiQUE sebuah kolom yang memiliki data yang berbeda atau tidak sama unique boleh 1,2,3 Dan seterusnya dan boleh tidak ada.

## Pertemuan 3



# Insert dan Select

Setelah mempelajari cara untuk membuat, melihat struktur dan menampilkan daftar table sekarang kita akan mempelajari cara memasukkan data pada table serta menampilkan hasil dari table tersebut, untuk melakukannya kita akan membaginya menjadi dua bagian yaitu *Insert* dan *Select*.

## Insert

query satu ini memiliki fungsi untuk memasukkan sebuah nilai pada table yang telah kita buat dan kita akan mempelajari insert data 1 baris dan lebih dari satu baris.

### Insert 1 Baris

untuk menginput data satu baris kita menggunakan format seperti berikut :

```
INSERT INTO [NAMA_TABLE]
VALUES (DATA_1, DATA_2, DATA_3, DATA_4);
```

atau untuk pengaplikasiannya seperti berikut :

```
MariaDB [rental_nero]> INSERT INTO PELANGGAN
->
-> VALUES (1, "Fachri", "Ramadhan", "083135219069");
Query OK, 1 row affected (0.064 sec)
```

### Insert Lebih 1 Baris

Sedangkan untuk menginput nilai yang lebih dari satu baris kita menggunakan format seperti dibawah ini :

```
INSERT INTO [NAMA_TABLE]
VALUES (DATA_1, DATA_2, DATA_3, DATA_4),
(DATA_1, DATA_2, DATA_3, DATA_4),
(DATA_1, DATA_2, DATA_3, DATA_4);
```

atau untuk pengaplikasiannya seperti berikut :

```
MariaDB [rental_nero]> INSERT INTO PELANGGAN
-> VALUES (2, "Chairil", "Abizali", '081214829012'),
-> (3,"Rahmat", "Ramadan", '081239171955');
Query OK, 2 rows affected (0.012 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

## Select

Selanjutnya query ini memiliki fungsi untuk menampilkan hasil dari table yang telah di inputkan (Insert) data kedalam tabel tersebut, berbeda dengan **desc** yang hanya menampilkan struktur dari table query ini menampilkan hasil dari table.

### Select all table

untuk menampilkan hasil dari seluruh table yang telah dibuat/menampilkan seluruh baris dan kolom kita menggunakan format seperti dibawah ini :

```
SELECT * FROM [NAMA_TABLE];
```

Dan hasilnya akan tampil seperti ini :

```
MariaDB [rental_nero]> SELECT * FROM PELANGGAN;
```

| Id_Pelanggan | Nama_Depan | Nama_Belakang | No_Telp      |
|--------------|------------|---------------|--------------|
| 1            | Fachri     | Ramadhan      | 083135219069 |
| 2            | Chairil    | Abizali       | 081214829012 |
| 3            | Rahmat     | Ramadan       | 081239171955 |

```
3 rows in set (0.001 sec)
```

## Analisis >

- Select ialah query yang berfungsi untuk menampilkan data pada table yang telah kita buat.
- Select ialah query yang berfungsi untuk menampilkan hasil table dan select ini terbagi menjadi 3 bagian.
- 3 Jenis Select ialah Select All Table, Select Field Spesifik dan Select kondisi atau "Where".
- " \* " simbol bintang ini memiliki makna "all" atau "semua"

**Kesimpulan :**\

## Select field spesifik

lalu untuk menampilkan beberapa kolom yang spesifik kita dapat menggunakan format yang sedikit berbeda dengan format all table, yaitu seperti dibawah ini :

```
SELECT NAMA_KOLOM_1, NAMA_KOLOM_2, NAMA_KOLOM_N FROM PELANGGAN;
```

Dan hasil yang akan tampil ialah kolom kolom yang di minta saja contoh dan hasilnya akan seperti ini :

```
MariaDB [rental_nero]> SELECT Id_Pelanggan, Nama_Depan, Nama_Belakang FROM PELANGGAN;
+-----+-----+-----+
| Id_Pelanggan | Nama_Depan | Nama_Belakang |
+-----+-----+-----+
|          1 | Fachri    | Ramadhan      |
|          2 | Chairil   | Abizali       |
|          3 | Rahmat    | Ramadan       |
+-----+-----+-----+
3 rows in set (0.001 sec)
```

## Select kondisi "where"

lalu kondisi yang saat satu ini berfungsi untuk mengambil data yang lebih spesifik dari sebuah field dengan simbol simbol aritmatika mulai dari "+", "-", "/", "%", ">", "<". Misalnya kita meminta untuk menampilkan field "Nama\_Depan" pada "Id\_Pelanggan" ke 2, kita dapat menggunakan simbol aritmatika seperti berikut :

```
SELECT Nama_Kolom FROM Nama_Table WHERE Id_Pelanggan=2;
```

Dan contoh serta hasilnya akan terlihat seperti berikut ini :

```
MariaDB [rental_nero]> SELECT Nama_Depan FROM PELANGGAN WHERE Id_Pelanggan=2;
+-----+
| Nama_Depan |
+-----+
| Chairil    |
+-----+
1 row in set (0.001 sec)
```

- Insert ialah query yang berfungsi untuk memasukkan data pada table yang telah kita buat.
- Select ialah query yang berfungsi untuk menampilkan hasil table dan select ini terbagi menjadi 3 bagian.
- 3 Jenis Select ialah Select All Table, Select Field Spesifik dan Select kondisi atau "Where".
- Where ini berisikan simbol simbol aritmatika mulai dari "+", "-", "/", "%", ">", "<".
- " \* " simbol bintang ini memiliki makna "all" atau "semua"

## Kesimpulan

Kesimpulannya ialah **insert** bertugas untuk memasukkan nilai pada table yang telah dibuat dan **Select** berfungsi untuk menampilkan hasil dari table yang telah dibuat dan di input datanya dari *Query* sebelumnya, lalu **Select** ini dapat menampilkan semua sesuai dengan yang kita menggunakan misalnya jika ingin menampilkan seluruh table kita menggunakan simbol " \* " atau All lalu jika ingin menampilkan beberapa field kita dapat menggunakan format hanya perlu memanggil nama fieldnya.

Lalu yang terakhir ialah kondisi "Where" dimana kita dapat memanggil nama field dengan menggunakan simbol aritmatika, misalnya kita ingin memanggil field "Nama\_Pelanggan" tapi hanya "Id\_Pelanggan" 2 kita dapat menggunakan format seperti ini `SELECT Nama_Kolom FROM Nama_Table WHERE Id_Pelanggan=2;`

## Update

Selanjutnya jika ingin mengganti nilai dari sebuah kolom tertentu kita bisa menggunakan *Query Update* lalu formatnya seperti dibawah ini :

```
Format :  
UPDATE [Nama_Table] SET [Nama_Kolom]="Nilai_Pengganti" WHERE kondisi;
```

Contoh :

```
UPDATE PELANGGAN SET No_Telp="083135219096" WHERE Id_Pelanggan=1;
```

Berikut ialah contoh pengaplikasian dan hasil dari penggunaan **Update** :

```
MariaDB [RENTAL_NERO]> UPDATE PELANGGAN SET No_Telp="083135219096" WHERE Id_Pelanggan=1;  
Query OK, 1 row affected (0.012 sec)  
Rows matched: 1  Changed: 1  Warnings: 0
```

## Delete

Kita juga dapat menghapus baris pada table dengan *Query Delete*, untuk menghapus keseluruhan baris kita dapat menggunakan format seperti ini :

Format :

```
DELETE FROM [Nama_Table] WHERE [Nama_Kolom];
```

Contoh :

```
DELETE FROM PELANGGAN WHERE Id_Pelanggan=6;
```

Berikut ialah contoh pengaplikasian dan hasil dari penggunaan **Delete** :

```
MariaDB [RENTAL_NERO]> DELETE FROM PELANGGAN WHERE Id_Pelanggan=6;  
Query OK, 1 row affected (0.012 sec)
```

```
MariaDB [RENTAL_NERO]> SELECT * FROM PELANGGAN;
```

| Id_Pelanggan | Nama_Depan | Nama_Belakang | No_Telp      |
|--------------|------------|---------------|--------------|
| 1            | Fachri     | Ramadhan      | 083135219096 |
| 2            | Chairil    | Abizali       | 081214829012 |
| 3            | Rahmat     | Ramadan       | 081239171955 |
| 4            | Abd        | Rahman        | 082135169816 |
| 5            | Rayhan     | Juli          | 084109156827 |

```
5 rows in set (0.000 sec)
```

## Analisis >

**Update** ialah *Query* untuk mengganti nilai yang telah ada pada sebuah table yang telah ada sebelumnya.

**Delete** ialah *Query* untuk menghapus baris pada sebuah tabel yang telah dibuat sebelumnya.

Penggunaan *Where* masih sangat berperan penting dalam kondisi seperti ini.

## Kesimpulan


dua *Query* yang akan dipelajari selanjutnya ialah untuk mengganti data dan menghapus baris data pada table. *Query* nya ialah **Update** untuk mengganti data yang telah ada pada table, dan *Query Delete* untuk menghapus nilai yang telah ada pada table yang telah kita buat. kedua *Query* ini memiliki format yang lumayan mirip, dimana memerlukan *Where* untuk menuliskan kondisinya.

Pisah file baru

## Pertemuan 4

# Select Lanjutan

Setelah mempelajari select di materi sebelumnya sekarang kita akan masuk ke dalam materi select lanjutan, fungsi dari select select ini ialah untuk mendapatkan hasil yang lebih spesifik dan lebih luas, sekarang kita akan mempelajari 7 select lanjutan (**AND ,OR ,BETWEEN-AND ,NOT BETWEEN ,<= ,>= ,<> ATAU !=**) Untuk penjelasan lebih lanjutnya ialah seperti berikut :

 Isi Table yang akan digunakan : >

| id_pelanggan | no_plat    | no_mesin | warna  | pemilik | peminjam | harga_rental |
|--------------|------------|----------|--------|---------|----------|--------------|
| 1            | DD 2650 XY | ACX3568  | Hitam  | Ibrahim | Afdal    | 50000        |
| 2            | DD 2440 AX | BCS1120  | Merah  | Ibrahim | Elia     | 100000       |
| 3            | B 1611 QC  | LSQ1112  | Silver | Baim    | Anty     | 50000        |
| 4            | DD 2901 JK | UQL1029  | Hitam  | Ibe     | NULL     | 150000       |
| 5            | DD 2210 LS | CJH1011  | Hitam  | Ibe     | NULL     | 100000       |

## AND

untuk **AND** ini akan mengambil "data 1" *dan* "data 2", contoh kodenya adalah seperti berikut :



```
SELECT warna,pemilik FROM mobil WHERE warna="Hitam" AND pemilik="ibrahim";
```

Dan hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT warna,pemilik FROM mobil WHERE warna="Hitam" AND pemilik="ibrahim";
+-----+-----+
| warna | pemilik |
+-----+-----+
| Hitam | Ibrahim |
+-----+-----+
1 row in set (0.001 sec)
```

## OR

Untuk **OR** ini akan mengambil "data 1" *atau* "data 2", contoh kodenya ialah seperti berikut :

```
SELECT warna,pemilik FROM mobil WHERE warna="Hitam" OR pemilik="ibrahim";
```

Dan hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT warna,pemilik FROM mobil WHERE warna="Hitam" OR pemilik="ibrahim";
+-----+-----+
| warna | pemilik |
+-----+-----+
| Hitam | Ibrahim |
| Merah | Ibrahim |
| Hitam | Ibe     |
| Hitam | Ibe     |
+-----+-----+
4 rows in set (0.000 sec)
```

## BETWEEN-AND

Untuk **BETWEEN-AND** ini akan mengambil antara "data 1" *sampai* "data 2" dibantu dengan **AND**, contoh kodenya ialah seperti berikut :

```
SELECT * FROM mobil WHERE harga_rental BETWEEN 100000 AND 200000;
```

Dan hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental BETWEEN 100000 AND 200000;
+-----+-----+-----+-----+-----+-----+-----+
| id_pelanggan | no_plat   | no_mesin | warna | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
|          2   | DD 2440 AX | BCS1120  | Merah | Ibrahim | Elia     |      100000  |
|          4   | DD 2901 JK | UQL1029  | Hitam | Ibe     | NULL     |      150000  |
|          5   | DD 2210 LS | CJH1011  | Hitam | Ibe     | NULL     |      100000  |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.000 sec)
```

## NOT BETWEEN

Untuk **NOT BETWEEN** ini akan mengambil "data" yang *bukan antara* "data 1" *dan* "data 2", contoh kodenya ialah seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental NOT BETWEEN 100000 AND 200000;
```

Dan hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental NOT BETWEEN 100000 AND 200000;
+-----+-----+-----+-----+-----+-----+-----+
| id_pelanggan | no_plat   | no_mesin | warna  | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
|          1   | DD 2650 XY | ACX3568  | Hitam  | Ibrahim | Afdal    |      50000   |
|          3   | B 1611 QC  | LSQ1112  | Silver | Baim    | Anty     |      50000   |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.006 sec)
```

<=

Untuk <= ini akan mengambil "data" lebih kecil atau sama dengan "nilai data", contoh kodenya ialah seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental <= 50000;
```

Dan hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental <= 50000;
+-----+-----+-----+-----+-----+-----+-----+
| id_pelanggan | no_plat   | no_mesin | warna  | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
|          1   | DD 2650 XY | ACX3568  | Hitam  | Ibrahim | Afdal    |      50000   |
|          3   | B 1611 QC  | LSQ1112  | Silver | Baim    | Anty     |      50000   |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.001 sec)
```

>=

Untuk >= ini akan mengambil "data" lebih besar atau sama dengan "nilai data", contoh kodenya ialah seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental >= 50000;
```

Dan hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental >= 50000;
+-----+-----+-----+-----+-----+-----+-----+
| id_pelanggan | no_plat   | no_mesin | warna  | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | DD 2650 XY | ACX3568 | Hitam | Ibrahim | Afdal    | 50000 |
| 2 | DD 2440 AX | BCS1120 | Merah | Ibrahim | Elia     | 100000 |
| 3 | B 1611 QC  | LSQ1112 | Silver | Baim    | Anty     | 50000 |
| 4 | DD 2901 JK | UQL1029 | Hitam | Ibe     | NULL     | 150000 |
| 5 | DD 2210 LS | CJH1011 | Hitam | Ibe     | NULL     | 100000 |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.001 sec)
```

## <> atau !=

Untuk <> atau != ini akan mengambil "data" yang tidak sama dengan "nilai data", contoh kodenya ialah seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental <> 50000;
```

Dan hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT * FROM mobil WHERE harga_rental <> 50000;
+-----+-----+-----+-----+-----+-----+-----+
| id_pelanggan | no_plat   | no_mesin | warna | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
|          2 | DD 2440 AX | BCS1120  | Merah | Ibrahim | Elia     |      100000 |
|          4 | DD 2901 JK | UQL1029  | Hitam | Ibe     | NULL     |      150000 |
|          5 | DD 2210 LS | CJH1011  | Hitam | Ibe     | NULL     |      100000 |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.000 sec)
```

## Tantangan

Untuk tantangan saya akan mengambil nama pemilik "Ibe" dengan cara memanggilnya dengan syarat nomor pelatnya yaitu "DD 2901 JK" lalu hasilnya akan seperti berikut :

```
MariaDB [rental_nero]> SELECT nama from akun WHERE password="12345";
+-----+
| nama |
+-----+
| Githa |
+-----+
1 row in set (0.001 sec)
```

### Analisis >

"AND" : Mengambil data 1 **dan** data 2.

"OR" : Mengambil data antara data 1 **atau** data 2.

"BETWEEN-AND" : Mengambil data **antara** data 1 **sampai** data 2.

"NOT BETWEEN" : Mengambil data yang tidak **antara** data 1 **sampai** data 2.

"<=" : Mengambil data yang lebih kecil atau sama dengan nilai data.

">=" : Mengambil data yang lebih besar atau sama dengan nilai data.

"<> atau !=" : Mengambil data yang **tidak** sama dengan nilai data.

Kesimpulan :

Select ini memiliki cakupan yang luas dan bervariasi semuanya juga memiliki kelebihan dan keunikan masing masing sehingga dapat menampilkan hasil sebuah nilai yang diinginkan, keberagaman select ini mulai dari AND ,OR ,BETWEEN-AND ,NOT BETWEEN ,<= ,>= ,<> ATAU !=.