

1. Pengenalan Basis data

DEFINISI BASIS

BASIS merupakan tempat,markas,gudang,dan tempat berkumpul.

DEFINISI DATA

DATA adalah sebuah informasi yang memiliki suatu objek seperti manusia,hewan,dll.

DEFINISI BASIS DATA

Basis data adalah sekumpulan data-data yang disimpan dalam komputer secara sistematis sehingga dapat diperiksa untuk mendapatkan sebuah informasi.

KESIMPULAN BASIS DATA

PERANAN BASIS DATA

Contoh peranan basis data di stasiun televisi **A**.menyimpan,mengelola konten, seperti 1.acara televisi,2.episode,3.video,4.iklan,5.penyiaran,6. Keuangan,7.penjualan iklan,dan 8.data karyawan,dan yang mengelolanya staf yang sudah di tentukan.

B.penyiaran dan siaran langsung di kelola oleh tim penyiar suatu stasiun televisi untuk mengatur jadwal program.

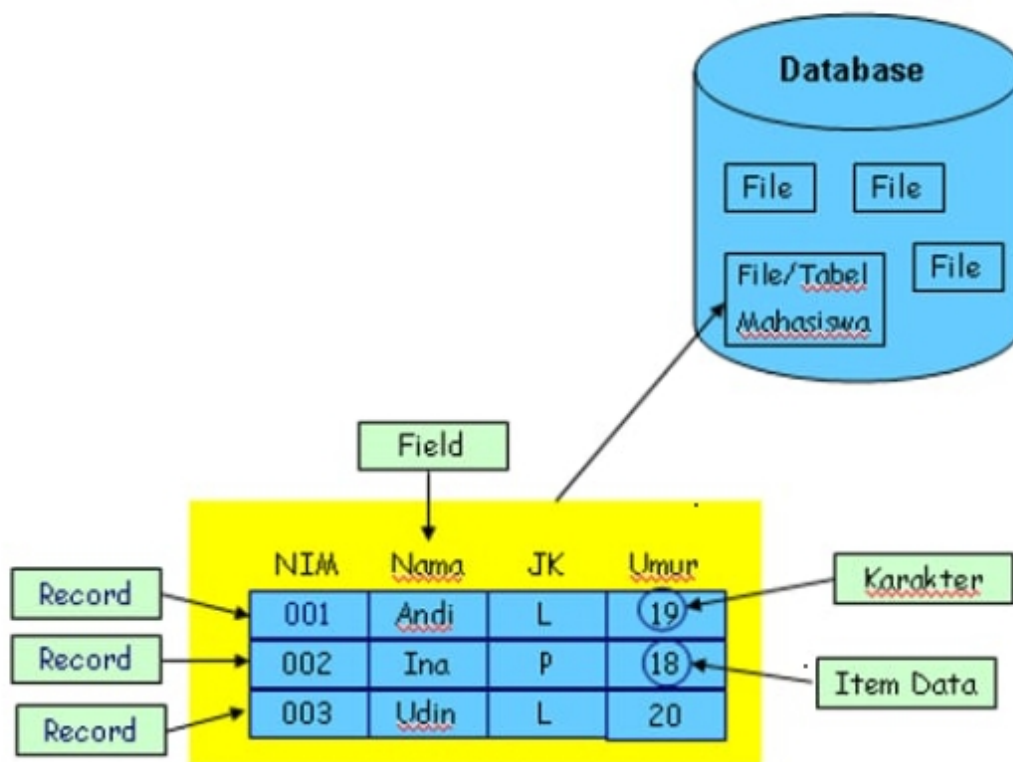
C.Berita dan Jurnalisme di kelola oleh tim jurnalis,reporter,editor,dan produser bekerja sama untuk menyampaikan berita terbaru.

D. Keuangan dan penjualan iklan di kelola oleh bendahara.

Selasa-16-january-2024

Tabel Database

[? Contoh Tabel Database >](#)



Struktur/Hirarki DataBase

Hirarki DataBase adalah struktur organisasi data dalam database yang mengatur hubungan antara entitas atau tabel data. Di dalam hirarki database, data diorganisir dalam bentuk pohon dengan satu entitas induk atau tabel utama yang memiliki beberapa entitas tabel yang terkait.

Record (Baris)

Record adalah kumpulan data terkait yang mewakili entitas tunggal atau item dalam basis data.

Field (Kolom)

Field adalah komponen terkecil dari sebuah record yang menyimpan nilai tunggal atau data spesifik. Field juga dikenal sebagai kolom dalam tabel basis data, dan setiap field dalam tabel mewakili atribut atau jenis data yang berbeda.

Item Data

Nilai spesifik di dalam setiap kolom. Contohnya, "Lionel Messi" adalah item data untuk kolom "Nama Pemain," "10" untuk "No Punggung," "Inter miami" untuk "Club' Pemain," dan

"36 Tahun" untuk "Umur Pemain."

Tabel Database Timnas Sepak bola Argentina

Pemain Argentina	No punggung	Klub Pemain	Umur
LIONEL MESSI	10	Inter Miami	36 tahun
R.DE PAUL	7	Atletico Madrid	29 tahun
PAULO DYBALA	21	AS Roma	30 tahun
L.PAREDES	5	AS Roma	29 tahun

2. Instalasi & Query awal Database

Selasa-23-jan-2024

Instalasi MySQL

Penggunaan Awal MySQL

Menggunakan Termux

1. Berikan akses termux ke memori internal .
`termux-setup-storage`
2. Muncul pop up untuk meminta izin akses ke memori internal.
klik izinkan/allow acces
3. Lakukan update dan sekaligus upgrade paket.
`pkg upgrade && upgrade -y`
4. Jika ada konfirmasi untuk melanjutkan instalasi. Silahkan klik y dan enter.
5. Install aplikasi MariaDB.
`pkg install mariadb`
6. Memberikan akses aman ke MySQL.
`mysql_safe`
7. Hentikan proses
`Ctrl + z`
8. Masuk ke dalam admin
`mysql -u root`

Referensi Video YouTube

<https://youtu.be/ez3nx3xH-y4?si=T4saycipqfBcqL1c>

Penggunaan Awal MySQL

Query

Struktur

```
mysql -u root
```

Contoh:

```
~ $ mysql -u root
mysql: Deprecated program name. It will be removed in a future release, use '/data
/data/com.termux/files/usr/bin/mariadb' instead
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 11.1.2-MariaDB MariaDB Server
```

Analisis & Kesimpulan MySQL -u root

MySQL

Ini adalah perintah untuk mengakses **shell MySQL**, yaitu antarmuka command-line untuk berinteraksi dengan server **MySQL**.

-U Root

Paramater ini menentukan pengguna (user) yang akan digunakan untuk masuk ke server MySQL. Dalam hal ini **root** adalah nama pengguna yang di berikan, dan **root** adalah tingkat tertinggi dengan hak akses penuh.

Kesimpulan

MySQL -u root memberikan akses penuh ke server MySQL dengan menggunakan **root**, yang memiliki hak akses maksimum. Penggunaan perintah ini perlu hati-hati untuk menghindari risiko keamanan.

DataBase

Membuat Database

Untuk membuat database di MySQL, anda dapat menggunakan perintah **CREATE DATABASE** dengan nama database yang akan di buat contohnya: **xi_rpl_1**; pastikan untuk memiliki hak akses Yang sesuai, dan verifikasi pembuatan database dengan perintah **SHOW DATABASE**.


Struktur

```
Create database nama database yang ingin dibuat;
```

Contoh:

```
Create database xi_rpl_1;
```

Hasil

```
MariaDB [(none)]> create database xi_rpl_1;   
Query OK, 1 row affected (0.006 sec)
```

```
MariaDB [(none)]> show databases;
```

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
| test |  
| xi_rpl_1 |  
+-----+
```

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

```
MariaDB [(none)]> █
```

Analisis:

`Create database` digunakan untuk membuat database. Contohnya seperti `create database xi_rpl_1`. `xi_rpl_1` adalah nama database yang akan kita buat.

Kesimpulan:

Jadi kalo kita ingin membuat sebuah database kita hanya perlu memasukkan kode `create database` dan nama databasenya (`xi_rpl_1`).

Menampilkan Database

Untuk menampilkan daftar database di MySQL, anda dapat menggunakan perintah **SQL SHOW DATABASE**. Perintah ini memberikan gambaran keseluruhan database yang

tersedia di server **MySQL**. pastikan pengguna yang digunakan memiliki izin untuk melihat database dan gunakan perintah melalui antarmuka **command-line** atau alat manajemen database seperti **phpMyAdmin**.


Struktur

```
Show databases;
```

Contoh

```
Show databases;
```

Hasil

```
MariaDB [(none)]> show databases;   
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
| test |  
| xi_rpl_1 |  
+-----+  
6 rows in set (0.009 sec)  
  
MariaDB [(none)]> █
```

Analisis:

Show Databases Berfungsi untuk menampilkan database yang kita buat.

Kesimpulan:

Jadi kesimpulannya **Show databases** itu berguna untuk menampilkan database yang di buat.

Menghapus Database

Untuk menghapus sebuah database di SQL, anda dapat menggunakan perintah **DROP DATABASE xi_rpl_1;** (Nama database yang akan di hapus). Namun perlu diingat bahwa tindakan ini akan menghapus semua data yang ada didalam database. Pastikan anda memiliki backup data yang dibutuhkan sebelum melanjutkan.

Struktur:

```
drop database nama_database;
```

Contoh:

```
drop database xi_rpl_1
```

```
MariaDB [(none)]> drop database xi_rpl_1  
Query OK, 0 rows affected (0.013 sec)
```

```
MariaDB [(none)]> show databases;
```

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
| test |  
+-----+  
5 rows in set (0.001 sec)
```

Analisis:

fungsi dari **drop database** adalah untuk menghapus sebuah database. **xi_rpl_1** nama database yang ingin dihapus, contohnya **drop database xi_rpl_1**. otomatis database dari **xi_rpl_1** akan terhapus. Jika ingin memastikan databasenya sudah terhapus kita ketik **==show database;**

Kesimpulan:

Jadi **drop database** itu berguna untuk menghapus database kita dengan menambahkan nama database yang ingin dihapus.

Menggunakan Database

Perintah **use** digunakan untuk beralih atau menggunakan sebuah database tertentu di server. Perintah ini sangat berguna ketika anda bekerja dengan beberapa database di server MySQL dan ingin fokus pada satu database dalam sesi tertentu. Contohnya: **use xi_rpl_1;**

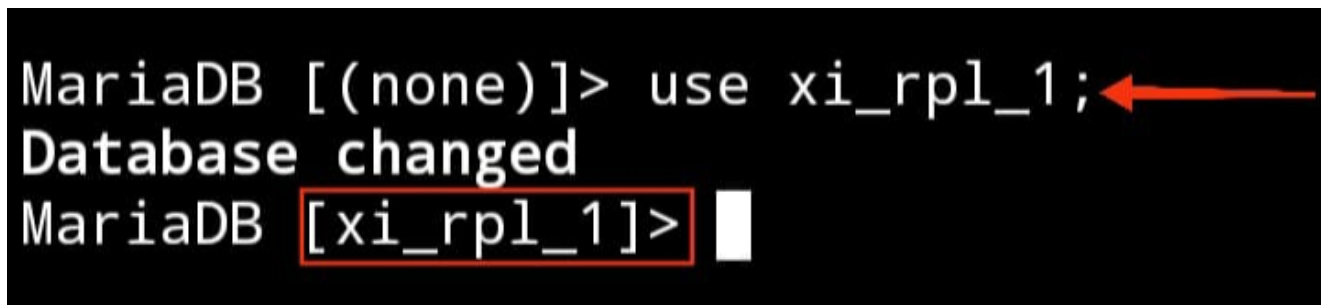
Struktur:

```
use nama_databasenya;
```

Contoh:

```
use xi_rpl_1;
```

Hasil



```
MariaDB [(none)]> use xi_rpl_1;  
Database changed  
MariaDB [xi_rpl_1]> █
```

Analisis:

use itu berfungsi untuk beralih database seperti **use xi_rpl_1** (nama database) maka akan beralih ke database xi_rpl_1.

Kesimpulan:

Jadi, **use** itu digunakan untuk beralih database.

Tipe Data

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.

Struktur:

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

Contoh:

```
MariaDB [xi_rpl_1]> show tables;
+-----+
| Tables_in_xi_rpl_1 |
+-----+
| contoh_tabel       |
+-----+
1 row in set (0.000 sec)

MariaDB [xi_rpl_1]> desc contoh_tabel;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id             | int(11)       | YES  |     | NULL    |       |
| harga          | decimal(10,2) | YES  |     | NULL    |       |
| jumlah_barang | tinyint(4)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.004 sec)

MariaDB [xi_rpl_1]> █
```

Analisis:

Tipe data **DECIMAL(10, 2)** menentukan angka desimal dengan presisi 10 digit dan skala 2 digit, yang berarti dapat menyimpan nilai hingga total 10 digit, dengan 2 digit di antaranya setelah koma desimal. **Tinyint** adalah tipe data yang dapat menampung nilai -128 sampai 127.

Kesimpulan

Jadi, kesimpulannya Tipe data **DECIMAL** itu akan menentukan bilangan desimal dan **TINYINT** itu untuk menampung sebuah nilai yang berukuran kecil.

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.

Struktur:

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

Contoh:

```
MariaDB [xi_rpl_1]> CREATE TABLE fatur_tabel (
->     nama CHAR(50),
->     alamat VARCHAR(100),
->     catatan TEXT,
->     status ENUM('Aktif', 'Non-Aktif')
-> );
```

Query OK, 0 rows affected (0.015 sec)

```
MariaDB [xi_rpl_1]> Show Tables;
```

```
+-----+
| Tables_in_xi_rpl_1 |
+-----+
| contoh_tabel       |
| fatur_tabel         |
+-----+
2 rows in set (0.001 sec)
```

```
MariaDB [xi_rpl_1]> desc fatur_tabel;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nama  | char(50) | YES | | NULL | |
| alamat | varchar(100) | YES | | NULL | |
| catatan | text | YES | | NULL | |
| status | enum('Aktif','Non-Aktif') | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.001 sec)
```

```
MariaDB [xi_rpl_1]> █
```

Analisis

- Kolom **nama** diubah menjadi VARCHAR(50) agar nama dapat maksimal 50 karakter. Kolom **alamat** juga diubah menjadi VARCHAR(100) untuk memperbolehkan maksimal 100 karakter untuk alamat.
- Batasan "NOT NULL" ditambahkan ke kolom "nama" dan "alamat" untuk memastikan bahwa kolom tersebut harus memiliki nilai dan tidak boleh kosong.
- Kolom **catatan** tetap sebagai TEKS, memungkinkan penyimpanan teks dalam jumlah lebih besar.
- **Status** adalah tipe ENUM dengan dua kemungkinan nilai: 'Aktif' dan 'Non-Aktif'.
- Batasan "NOT NULL" juga ditambahkan ke kolom "status" untuk memastikan bahwa kolom tersebut harus memiliki nilai.

Kesimpulan

Dalam contoh tersebut, **nama** menggunakan tipe data **char** dengan panjang tetap, **alamat** menggunakan tipe data **VARCHAR** dengan panjang variabel, **catatan** menggunakan tipe data **TEXT** untuk menyimpan teks yang mungkin panjangnya bervariasi, dan **status** menggunakan tipe data **ENUM** untuk membatasi nilai yang mungkin.

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.

Struktur

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

hasil

```

MariaDB [xi_rpl_1]> CREATE TABLE ftur_Tabel (
  ->     tanggal DATE,
  ->     waktu TIME,
  ->     datetimekolom DATETIME,
  ->     timestampkolom TIMESTAMP
  -> );
Query OK, 0 rows affected (0.003 sec)

MariaDB [xi_rpl_1]> show tables;
+-----+
| Tables_in_xi_rpl_1 |
+-----+
| ContohTabel        |
| contoh_tabel        |
| fathur_Tabel        |
| fatur_Tabel         |
| fatur_tabel         |
| fthur_Tabel         |
| ftur_Tabel          |
+-----+
7 rows in set (0.001 sec)

MariaDB [xi_rpl_1]> desc ftur_table;
ERROR 1146 (42S02): Table 'xi_rpl_1.ftur_table' doesn't exist
MariaDB [xi_rpl_1]> desc ftur_Tabel;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| tanggal   | date      | YES  |     | NULL    |       |
| waktu      | time      | YES  |     | NULL    |       |
| datetimekolom | datetime | YES  |     | NULL    |       |
| timestampkolom | timestamp | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.006 sec)

MariaDB [xi_rpl_1]> 

```

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, pada dasarnya, 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`.

Tipe Data Pilihan

ENUM

fungsinya untuk menggambarkan sebuah nilai terbatas atau terdefinisi.

SET

fungsinya untuk menyimpan urutan nilai yang tidak berurut.

Tabel

Buat Tabel

Struktur

```
CREATE TABLE [nama_tabel] (  
  Nama_kolom1 tipe_data(ukuran) (tipe_constraint) ,  
  Nama_kolom2 tipe_data(ukuran) (tipe_constraint) ,  
  Nama_kolom3 tipe_data(ukuran) (tipe_constraint) ,  
);
```

Contoh

```
CREATE TABLE pelanggan (  
  id_pelanggan int(4)PRIMARY KEY NOT NULL , nama_depan varchar(25) NOT NULL ,  
  nama_belakang varchar(25) NOT NULL , no_telp char(12)UNIQUE  
);
```

Hasil

```
MariaDB [(none)]> CREATE TABLE pelanggan (  
  -> id_pelanggan int(4)PRIMARY KEY NOT NULL , nama_depan varchar  
  r(25) NOT NULL , nama_belakang varchar(25) NOT NULL , no_telp char  
  (12)UNIQUE
```

Analisis

Id pelanggan adalah kolom bilangan bulat dengan panjang 4 digit.ditetapkan sebagai kunci utama tabel dan tidak boleh **NULL**.

nama depan kolom stringnya panjang 25 karakter maksimumnya.

nama belakang kolom stringnya panjang 25 karakter maksimumnya.

no telp kolom karakter yang panjangnya 12 karakter dan memiliki batas UNIQUE yang artinya setiap nilai di kolom harus unik.

Kesimpulan

Struktur Tabel

Struktur

```
Desc [nama_tabel];
```

Contoh

```
desc pelanggan;
```

Hasil

```
MariaDB [rental_fatur]> desc pelanggan;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_pelanggan   | int(4)         | NO   | PRI | NULL    |       |
| nama_depan     | varchar(25)    | NO   |     | NULL    |       |
| nama_belakang  | varchar(25)    | YES  |     | NULL    |       |
| no_telp        | char(12)       | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.002 sec)
```

Analisis

`Desc pelanggan` berguna untuk menampilkan struktur dari database yang sudah di buat dan `pelanggan` itu adalah nama tabel yang ingin di tampilkan.

Kesimpulan

Jadi, `desc` itu untuk menampilkan struktur database.

Menampilkan Daftar Tabel

Struktur

```
Show tables;
```

Hasil

```
MariaDB [rental_fatur]> show tables;
+-----+
| Tables_in_rental_fatur |
+-----+
| pelanggan              |
+-----+
1 row in set (0.000 sec)
```

Analisis

Show tables berfungsi untuk membuka daftar tabel yang sudah kita buat. maka semua isi dari tabel akan tampil.

Kesimpulan

Show tables itu untuk membuat isi dari tabel

QNA

❓ Perbedaan PRIMERY KEY dan UNIQUE! >

- **Premery** Key bertugas membedakan nilai yang ada pada tabel seperti NIS.
- **UNIQUE** bertugas untuk memastikan bahwa tidak ada nilai duplikat dalam kolom tersebut. contohnya seperti, no wa, dan alamat email.

❓ Mengapa Hanya Kolom Id Pelanggan yang menggunakan Constraint "PRIMARY KEY"? >

Karena dapat memastikan integritas data dengan mencegah duplikasi dan memastikan memiliki entitas yang jelas.

❓ Mengapa pada kolom no_telp yang menggunakan tipe data CHAR bukan VARCHAR? >

Karena tipe data **CHAR** dapat menyimpan setiap nilai yang panjang dengan tetap.

❓ Mengapa Hanya kolom no_telp yang menggunakan constraint "UNIQUE"? >

Karena dengan menggunakan constraint "**UNIQUE**" kita dapat memastikan bahwa setiap no telp yang dimasukkan ke tabel hanya muncul sekali.

❓ Mengapa kolom no_telp tidak memakai constraint "NOT NULL", sementara kolom lainnya Menggunakan constraint tersebut? >

No telp di anggap opsional, nomor telpon menjadi wajib saat pengguna saat pengguna melakukan langkah-langkah tertentu.

Insert dan Select

Insert

Query ini berfungsi untuk memasukkan nilai pada dalam tabel yang sudah di buat.

Insert 1 Baris

Struktur

```
Insert into [nama_table]
Values (data_1,data_2,data_3,data_4);
```

Contoh

```
Insert into pelanggan
Values (1,"Fatur","rahman",'08960214172');
```

Hasil

```
MariaDB [rental_fatur]> Insert into pelanggan
-> Values (1,"Fatur","rahman",'08960214172');
Query OK, 1 row affected (0.009 sec)

MariaDB [rental_fatur]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp      |
+-----+-----+-----+-----+
|          1 | Fatur      | rahman         | 08960214172 |
+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [rental_fatur]> █
```

Analisis

`insert into pelanggan` adalah tabel yang kita ingin isi dengan beberapa data contohnya data 1 itu untuk nomor urut (1),data 2 untuk nama depan (Fatur), data 3 untuk nama belakang(Rahman),dan data ke empat untuk no telp(08960214172).

Kesimpulan

Kesimpulannya adalah kalo kita ingin menambahkan isi dari tabel cukup masukkan kode `insert into` serta nama tabelnya (pelanggan).

Insert lebih 1 Baris

Struktur

```
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);
```

Contoh

```
INSERT INTO pelanggan Values
(2, "muh", "taufik", '087998216'), (3, "ahsan", "putra", '083673749'),
(4, "muh", "zhafran", '081372347');
```

Hasil

```
MariaDB [rental_fatur]> INSERT INTO pelanggan Values
-> (2, "muh", "taufik", '087998216'), (3, "ahsan", "putra", '083673749'), (4, "muh", "zhafran", '081372347');
Query OK, 3 rows affected (0.005 sec)
Records: 3 Duplicates: 0 Warnings: 0

MariaDB [rental_fatur]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | Fatur | rahman | 08960214172 |
| 2 | muh | taufik | 087998216 |
| 3 | ahsan | putra | 083673749 |
| 4 | muh | zhafran | 081372347 |
+-----+-----+-----+-----+
4 rows in set (0.000 sec)

MariaDB [rental_fatur]> █
```

Analisis

Caranya Sama seperti insert 1 baris tetapi ini memasukkan lebih dari 1 baris data di dalam tabel.

Kesimpulan

Kesimpulannya kalo kita ingin menambahkan isi tabel lebih dari 1 Baris itu cukup masukkan data-datanya secara berurutan menggunakan `insert into nama tabel (pelanggan)`.

Select

query ini berfungsi menampilkan hasil dari table yang telah kita input (Insert) data kedalam tabel tersebut.

Select all table

Select all table berfungsi untuk menampilkan hasil table yang telah dibuat.

Struktur

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

Contoh

```
SELECT * FROM pelanggan;
```

Hasil

```
MariaDB [rental_fatur]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | Fatur | rahman | 08960214172 |
| 2 | muh | taufik | 087998216 |
| 3 | ahsan | putra | 083673749 |
| 4 | muh | zhafran | 081372347 |
+-----+-----+-----+-----+
4 rows in set (0.048 sec)
```

Analisis

`Select * from` adalah untuk membuka isi dari tabel yang sudah kita buat dengan menambahkan nama tabelnya (mobil)

Kesimpulan

Kesimpulannya `select * from` berguna untuk menampilkan tabel yang sudah dibuat.

Select field spesifik

Select field spesifik itu menampilkan beberapa kolom yang spesifik kita dapat menggunakan format yang sedikit berbeda dengan format all table, yaitu seperti dibawah ini

Struktur

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

Contoh

```
SELECT nama_depan, nama_belakang, no_telp FROM pelanggan;
```

Hasil

```
MariaDB [rental_fatur]> SELECT nama_depan, nama_belakang, no_telp FROM pelanggan;
+-----+-----+-----+
| nama_depan | nama_belakang | no_telp |
+-----+-----+-----+
| Fatur      | rahman        | 08960214172 |
| muh        | taufik        | 087998216   |
| ahsan      | putra         | 083673749   |
| muh        | zhafran       | 081372347   |
+-----+-----+-----+
4 rows in set (0.001 sec)

MariaDB [rental_fatur]> █
```

Analisis

Select adalah kata kunci yang digunakan untuk mengambil kolom-kolom tertentu dari tabel.

nama_depan, nama_belakang, no_telp adalah kolom-kolom yang dipilih untuk ditampilkan dalam hasil query. FROM pelanggan menunjukkan bahwa data diambil dari tabel bernama 'pelanggan'. 'pelanggan' adalah nama tabel.

Kesimpulan

kesimpulannya adalah query ini akan menghasilkan hasil yang terdiri dari kolom-kolom nama_depan, nama_belakang, dan no_telp dari tabel pelanggan.

Select kondisi "where"

Struktur

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

Contoh

```
SELECT nama_depan FROM pelanggan WHERE Id_Pelanggan=2;
```

Hasil

```
MariaDB [rental_fatur]> SELECT nama_depan FROM pelanggan WHERE Id_Pelanggan=2;
+-----+
| nama_depan |
+-----+
| muh        |
+-----+
1 row in set (0.015 sec)

MariaDB [rental_fatur]> █
```

Analisis

`select` adalah kata kunci yang digunakan untuk memilih kolom-kolom tertentu yang akan ditampilkan dalam hasil query.

`nama_depan` Ini adalah kolom yang dipilih untuk ditampilkan dalam hasil query. `from pelanggan` adalah nama tabelnya, dan `where id_pelanggan=2` nomor kolomnya contohnya kolom nomor 2 berarti yang dipanggil nama depan dari kolom nomor 2.

Kesimpulan

Jadi kesimpulannya adalah bahwa query ini akan mengambil nama depan dari pelanggan yang memiliki `Id_Pelanggan` sama dengan 2 dari tabel `pelanggan`.

Update

Struktur

```
Mysql > UPDATE nama_tabel SET nama_kolom WHERE kondisi;
```

Contoh

```
Update pelanggan SET nama_belakang="Rahman" WHERE id_pelanggan="1";
```

Hasil

```

MariaDB [rental_fatur]> Update pelanggan SET nama_belakang="Rahman
" WHERE id_pelanggan="1";
Query OK, 1 row affected (0.006 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [rental_fatur]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
|          1 | Fatur      | Rahman        | 0871584817 |
+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [rental_fatur]> █

```

Analisis

Update bertugas untuk mengupdate sebuah tabel contohnya `update (pelanggan)` nama tabelnya, SET itu berfungsi untuk menyimpan satu atau lebih nilai dari himpunan yang telah ditentukan (`nama_belakang`) bagian kolom tabel yang ingin diubah dan (`rahman`) hasilnya ketika kita sudah mengubahnya.

Kesimpulan

Jadi, kalo kita ingin mengubah/update tabel kita kita hanya perlu memasukkan kode `update` serta nama tabel, nama kolom tabel yang ingin kita ubah.

Delete

Struktur

```
mysql > DELETE FROM nama_tabel WHERE kondisi;
```

Contoh

```
DELETE FROM pelanggan WHERE id_pelanggan="3";
```

Hasil

```

MariaDB [rental_fatur]> DELETE FROM pelanggan WHERE id_pelanggan=
"3";
Query OK, 1 row affected (0.005 sec)

MariaDB [rental_fatur]> select + from pelanggan;
ERROR 1064 (42000): You have an error in your SQL syntax; check th
e manual that corresponds to your MariaDB server version for the r
ight syntax to use near 'from pelanggan' at line 1
MariaDB [rental_fatur]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
|          1 | Fatur     | Rahman        | 0871584817 |
|          2 | muh       | zhafran       | 089658420030 |
+-----+-----+-----+-----+
2 rows in set (0.000 sec)

```

Analisis

Delete itu berfungsi untuk menghapus kolom dari tabel, (pelanggan) nama tabelnya, dan WHERE id_pelanggan="3"; adalah nama kolom dari tabel, Contohnya (id_pelanggan="3") adalah bagian kolom tabel yang ingin kita hapus, (3) berarti kolom no 3 yang kita ingin hapus.

Kesimpulan

Untuk menghapus sebuah kolom tabel kita masukkan kode delete dan nama kolom tabel serta kolom ke berapa yang ingin kita hapus.

Delete tabel

Struktur

```
drop table nama_table;
```

Contoh

```
drop table pesanan;
```

Hasil


```

MariaDB [rental_fatur]> show tables;
+-----+
| Tables_in_rental_fatur |
+-----+
| Pesanan                |
| pelanggan              |
+-----+
2 rows in set (0.001 sec)

MariaDB [rental_fatur]> drop table pesanan;
ERROR 1051 (42S02): Unknown table 'rental_fatur.pesanan'
MariaDB [rental_fatur]> drop table pelanggan;
Query OK, 0 rows affected (0.010 sec)

MariaDB [rental_fatur]> show tables;
+-----+
| Tables_in_rental_fatur |
+-----+
| Pesanan                |
+-----+
1 row in set (0.000 sec)

```

Analisis

`Drop tabel` berfungsi untuk menghapus sebuah tabel dan (`pesanan`) nama tabelnya yang ingin kita hapus.dan untuk mengecek apakah tabelnya sudah terhapus masukkan kode `show tables`

Kesimpulan

Jadi,kalo kita ingin hapus tabel kita masukkan kode `drop table` serta nama tabel yang ingin di hapus.