

# **Belajar MySQL Dari NOL**

**Isa Hamdan, S.Kom**

## **Kenapa harus menggunakan database?**

Kebanyakan aplikasi yang dibuat digunakan untuk menyimpan dan melihat data. Data adalah bagian terkecil yang digunakan sebagai penyusun informasi. Karena fungsi yang sangat penting dari data agar dapat digunakan sebagai sumber informasi yang benar. Maka perlu dibuat sebuah rumah data yang baik dan benar. Rumah data inilah yang disebut dengan **DATABASE**.

Desain database yang baik akan membuat database tersebut dapat berjalan dengan baik dan sesuai dengan harapan ketika database dirancang dan dibuat. Keterampilan membuat database akan sangat menentukan berhasil tidaknya aplikasi dibuat. Database adalah jantung dari aplikasi.

Proses pembuatan aplikasi selalu dimulai dari desain database yang benar dulu baru kemudian ke desain tampilan (*USER INTERFACE*), baru kemudian ke *CODING* atau pembuatan kode program. Buku ini akan memandu untuk proses pembuatan database menggunakan MySQL.

Terimakasih sudah membaca dan menggunakan buku ini sebagai bahan untuk belajar. Jika ada masukan kritik atau saran demi perbaikan isi silahkan kirim ke email :  
[smkrevit@gmail.com](mailto:smkrevit@gmail.com)

Untuk kedua orang tua (ibu & bapak), buku ini Saya buat sebagai  
do'a anak kepada kedua orang tua, seperti sabda Nabi Kita.

## **DAFTAR ISI**

DASAR PENGGUNAAN.....	7
DASAR PENGGUNAAN.....	8
INSTALASI XAMPP .....	8
KELUAR DARI MYSQL .....	9
MEMERIKSA MYSQL JALAN ATAU TIDAK .....	10
CEK VERSI MYSQL.....	10
MENAMPILKAN HELP MYSQL.....	10
MENAMPILKAN TANGGAL .....	11
MENAMPILKAN WAKTU ATAU JAM.....	11
MENGUNAKAN KALKULATOR.....	11
DDL.....	13
DDL (Data Definition Language).....	14
MENAMPILKAN DATABASE .....	14
MEMBUAT DATABASE.....	15
MENGHAPUS DATABASE.....	15
MENGAKTIFKAN DATABASE.....	16
TIPE DATA.....	16
MEMBUAT TABEL.....	19
MENAMPILKAN TABEL .....	20
MENAMPILKAN STRUKTUR TABEL.....	20
MENAMBAH KOLOM.....	22
MERUBAH TIPE DATA.....	22
MEMBERI NILAI DEFAULT PADA KOLOM .....	23
MENGHAPUS KOLOM .....	24
MERUBAH NAMA KOLOM.....	25
MENAMBAHKAN PRIMARY KEY .....	26
MENAMPILKAN <i>ENGINE</i> YANG DIGUNAKAN .....	26
MENAMBAH KOLOM SETELAH KOLOM .....	28
MEMBUAT INDEX.....	29
MENAMPILKAN INDEX .....	30

MENGHAPUS INDEX.....	30
DML.....	31
<i>Data Manipulation Language</i> .....	31
INSERT SEMUA KOLOM.....	32
INSERT SEBAGIAN KOLOM.....	33
DELETE SEBAGIAN RECORD (BARIS DATA).....	33
DELETE SEMUA RECORD .....	34
UPDATE SEBAGIAN RECORD .....	34
UPDATE SEMUA RECORD.....	35
TABEL MASTER DAN TABEL TRANSAKSI (DETAIL) .....	36
TABEL MASTER.....	36
TABEL TRANSAKSI.....	36
PRIMARY KEY DAN FOREIGN KEY .....	36
RELATIONAL DEPENDENCIES (HUBUNGAN KETERGANTUNGAN).....	37
PROSES BISNIS (ALUR KERJA) .....	37
MENGUJI HASIL PEMBUATAN RELASI.....	40
PEMBUATAN VIEW.....	43
MENAMPILKAN SEMUA VIEW YANG SUDAH DIBUAT .....	44
MELIHAT ISI VIEW .....	44
MENGHAPUS VIEW .....	44
SELECT SEMUA KOLOM (*) .....	45
SELECT SEBAGIAN KOLOM .....	45
SELECT ORDER.....	46
SELECT GROUP .....	47
PENGUJIAN WHERE.....	47
SUBQUERY (SELECT IN SELECT) .....	49
MEMBUAT RELASI ANTAR TABEL.....	51
RELASI 2 TABEL MASTER DAN 1 TABEL TRANSAKSI .....	52
DUMMY DATA.....	53
KONSEP TRIGGER .....	54
PEMBUATAN TRIGGER .....	59

MENAMPILKAN TRIGGER .....	61
PENGUJIAN TRIGGER .....	61
HAPUS TRIGGER .....	65
JOIN (GABUNGAN TABEL) .....	66
INNER JOIN (MENGAMBIL BAGIAN YANG ADA DI TABEL MASTER DAN TABEL TRANSAKSI) .....	68
LEFT JOIN (MENAMPILKAN TABEL MASTER).....	70
RIGHT JOIN (MENAMPILKAN TABEL TRANSAKSI) .....	70
STORE PROCEDURE .....	71
MENAMPILKAN SEMUA PROCEDURE .....	72
MENGHAPUS STORE PROCEDURE .....	72
FUNCTION .....	73
MENAMPILKAN SEMUA FUNCTION .....	77
MENGHAPUS FUNCTION.....	77
SELECT AGGREGATE .....	78
SELECT BETWEEN (SELECT ANTARA DUA NILAI) .....	79
SELECT DISTINCT (MENAMPILKAN DATA YANG SAMA HANYA SATU KALI).....	80
START TRANSACTION, COMMIT, DAN ROLLBACK.....	80
DCL .....	<b>Error! Bookmark not defined.</b>
<i>Data Control Language</i> .....	<b>Error! Bookmark not defined.</b>
TENTANG DCL (Data Control Language) .....	85
MENAMPILKAN SEMUA USER.....	85
MENAMBAH USER .....	86
MEMBERIKAN PASSWORD PADA USER .....	86
MENGUJI USER DAN PASSWORD YANG TELAH DIBUAT.....	87
MEMBERIKAN HAK AKSES ( <i>PRIVILEGES</i> ) USER KE DATABASE.....	87
MENAMPILKAN USER YANG SEDANG LOGIN (MASUK) .....	88
MENAMPILKAN HAK AKSES YANG DIBERIKAN.....	89
MENGHAPUS HAK AKSES .....	89
MEMBERIKAN HAK AKSES PADA DATABASE DENGAN TABEL .....	90
MEMBERIKAN HAK AKSES PADA SELECT, INSERT, DELETE, UPDATE, PADA TABEL .....	92
UBAH PASSWORD USER.....	94

HAPUS USER.....	94
MERUBAH PASSWORD USER [root] .....	95
DAFTAR PUSTAKA .....	96

# **DASAR PENGGUNAAN**

## DASAR PENGGUNAAN

Sebelum menggunakan MySQL, pastikan MySQL sudah ter install di komputer atau laptop. Agar latihan anda sama dengan yang ada dibuku. Sebaiknya ikuti aplikasi yang digunakan di buku.

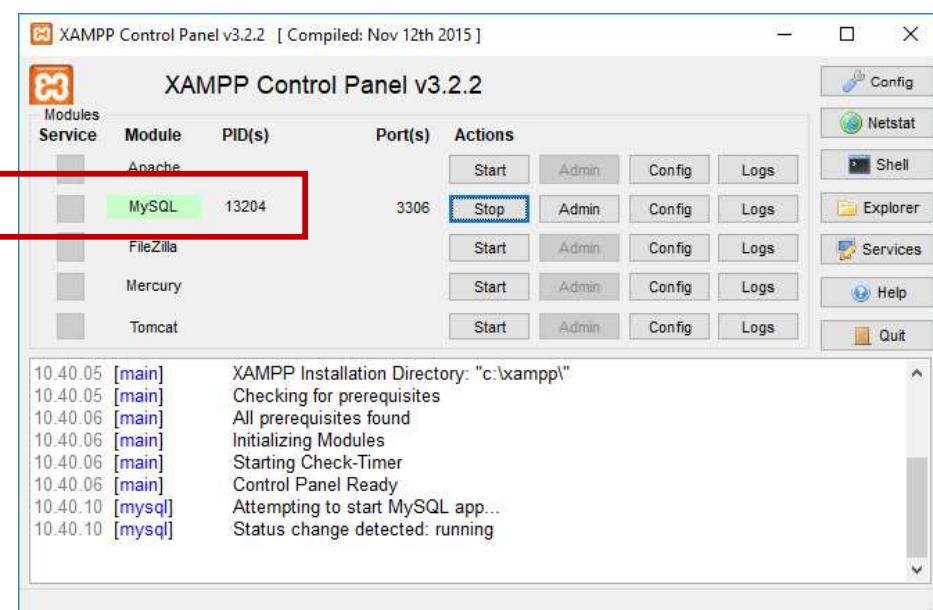
## INSTALASI XAMPP

Aplikasi yang digunakan sebagai bahan belajar adalah XAMPP. Download XAMPP di lokasi sebagai berikut:

<https://www.apachefriends.org/download.html>

install XAMPP di di drive C atau di D laptop atau komputer anda.

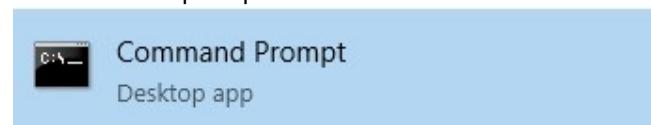
Jalankan XAMPP menggunakan XAMPP control. Pastikan service dari MySQL berjalan dengan baik.



Setelah XAMPP berhasil dijalankan maka buka aplikasi Command Prompt di windows dengan cara ketik [cmd] di pencarian start windows 10.



Pilih command prompt



```
cmd Command Prompt
Microsoft Windows [Version 10.0.16299.251]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\komputerkit>
```

Setelah terbuka jalankan MySQL dengan mengetik

```
C:\Users\komputerkit>cd C:\xampp\mysql\bin
```

Lokasi C atau D sesuai dengan XAMPP yang ada di komputer anda. XAMPP yang ada di buku diletakan di **C. Jalankan MySQL dengan mengetik**

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
```

Karena password default atau bawaan XAMPP nya kosong, langsung ketik enter saja.  
Jika MySQL berjalan maka akan tampil

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.30-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

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

## KELUAR DARI MYSQL

Untuk keluar dari MySQL gunakan perintah;

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

## **MEMERIKSA MYSQL JALAN ATAU TIDAK**

Untuk mengecek MySQL berjalan atau tidak, pastikan keluar dari MySQL dulu kemudian gunakan perintah dibawah. Tekan enter jika passwordnya kosong;

```
C:\xampp\mysql\bin>mysqladmin -u root -p ping  
Enter password:
```

## **CEK VERSI MYSQL**

Kita bisa mengecek versi MySQL yang digunakan menggunakan perintah;  
Pastikan sudah keluar dari MySQL sebelum mengecek versinya.

```
C:\xampp\mysql\bin>mysqladmin -u root -p version  
Enter password:
```

Tampilan hasil pengecekan versi MySQL

```
C:\xampp\mysql\bin>mysqladmin -u root -p version  
Enter password:  
mysqladmin Ver 9.1 Distrib 10.1.30-MariaDB, for Win32 on AMD64  
Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.  
  
Server version      10.1.30-MariaDB  
Protocol version   10  
Connection          localhost via TCP/IP  
TCP port            3306  
Uptime:
```

## **MENAMPILKAN HELP MYSQL**

Sangat sulit mengingat semua perintah MySQL, karena itu dibutuhkan bantuan agar kita bisa mengetahui perintah MySQL yang akan digunakan.

Untuk melihat bantuan MySQL, anda harus masuk dulu ke MySQL dengan perintah;

```
C:\xampp\mysql\bin>mysql -u root -p  
Enter password:
```

Tanda jika anda sudah ada dalam MySQL adalah;

```
MariaDB [(none)]> -
```

Untuk menampilkan HELP bisa menggunakan perintah

```
MariaDB [(none)]> help    atau MariaDB [(none)]> ?
```

## MENAMPILKAN TANGGAL

Gunakan perintah berikut untuk menampilkan tanggal. Pastikan akhir perintah di beri tanda **TITIK KOMA** (**;**) kemudian tekan enter.

```
MariaDB [(none)]> select curdate();
+-----+
| curdate() |
+-----+
| 2018-03-13 |
+-----+
1 row in set (0.00 sec)
```

## MENAMPILKAN WAKTU ATAU JAM

```
MariaDB [(none)]> select curtime();
+-----+
| curtime() |
+-----+
| 11:17:03 |
+-----+
1 row in set (0.00 sec)
```

## MENGGUNAKAN KALKULATOR

Untuk menggunakan kalkulator di MySQL gunakan perintah sebagai berikut;

### Penjumlahan

```
MariaDB [(none)]> select 3 + 4;
+-----+
| 3 + 4 |
+-----+
|      7 |
+-----+
1 row in set (0.00 sec)
```

### Perkalian

```
MariaDB [(none)]> select 3 * 4;
+-----+
| 3 * 4 |
+-----+
|     12 |
+-----+
1 row in set (0.00 sec)
```

### Pengurangan

```
MariaDB [(none)]> select 3 - 4;
+-----+
| 3 - 4 |
+-----+
|      -1 |
+-----+
1 row in set (0.00 sec)
```

### Pembagian

```
MariaDB [(none)]> select 3 / 4;
+-----+
| 3 / 4 |
+-----+
| 0.7500 |
+-----+
1 row in set (0.00 sec)
```

### Modulo (Sisa hasil pembagian)

```
MariaDB [(none)]> select 5 % 2;
+-----+
| 5 % 2 |
+-----+
|      1 |
+-----+
1 row in set (0.00 sec)
```

### Hasil pembagian integer (utuh)

```
MariaDB [(none)]> select 5 div 3;
+-----+
| 5 div 3 |
+-----+
|      1 |
+-----+
1 row in set (0.00 sec)
```

# **DDL**

*Data Definition Language*  
**Pembuatan Database, Tabel, & Index**

## DDL (Data Definition Language)

DDL adalah proses pembuatan RUMAH dari data. Data akan disimpan pada KOLOM, kolom disimpan di TABEL. Tabel disimpan di DATABASE.

Untuk latihan kita akan membuat database **APLIKASI TOKO**. Desain dari database toko sebagai berikut;



UNTUK CARA DESAIN DATABASE ADA DI MATERI BERBEDA DAN TIDAK ADA DI DALAM BUKU INI.

## MENAMPILKAN DATABASE

Untuk menampilkan semua database yang ada di MySQL komputer anda. Gunakan perintah;

```
SHOW DATABASES;
```

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

Hasil perintah diatas adalah;

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| dblaravel
| dbsekolah
| information_schema
| latihan
| mysql
| performance_schema
| phpmyadmin
| test
| tutorialci
+-----+
9 rows in set (0.00 sec)
```

## MEMBUAT DATABASE

Pada latihan ini, akan membuat database dengan nama dbtoko. Gunakan perintah berikut;

```
CREATE DATABASE nama_database;
```

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

Periksa database yang sudah dibuat dengan menggunakan perintah;

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| dblaravel
| dbsekolah
| dbtoko
| information_schema
| latihan
| mysql
| performance_schema
| phpmyadmin
| test
| tutorialci
+-----+
10 rows in set (0.00 sec)
```

## MENGHAPUS DATABASE

Jika ada database yang tidak digunakan bisa dihapus dengan perintah;

```
DROP DATABASE nama_database;
```

```
MariaDB [dbtoko]> drop database dbtoko;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> -
```

## MENGAKTIFKAN DATABASE

Jika database sebelumnya dihapus buat lagi dengan nama [dbtoko], gunakan perintah diatas. Sebelum membuat tabel anda harus memilih atau mengaktifkan database yang akan digunakan sebagai tempat penyimpanan tabel. Untuk memilih atau mengaktifkan gunakan perintah sebagai berikut;

```
USE nama_database;
```

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

Perhatikan terjadi perubahan dari [none] menjadi [dbtoko] ini menunjukan bahwa database yang aktif yang akan digunakan adalah [dbtoko]

## TIPE DATA

sebelum membuat tabel anda harus mengetahui tipe data yang akan digunakan di dalam pembuatan tabel.

### TIPE DATA NUMERIK

Tipe data numerik berisi sekumpulan tipe data sejenis yang mampu menangani data-data numerik. Berikut ini beberapa tipe data yang digunakan dalam MySQL.

Type	Bytes	Keterangan
TINYINT	1	Type ini merupakan bentuk numerik yang paling kecil dalam menangani data di dalamnya, yang dapat menangani data mulai dari angka -128 sampai dengan 127.
SMALLINT	2	Memiliki kemampuan menyimpan data lebih besar dari TINYINT, yaitu mampu untuk menangani data mulai dari -32768 sampai dengan 32767.
MEDIUMINT	3	Mampu menangani data mulai dari -8388608 sampai dengan 8388607.
INT	4	Type INT merupakan type yang sangat sering digunakan dalam pembuatan database, karena type ini dirasakan sangat cukup menampung dalam menangani data, yaitu mampu menyimpan data mulai dari -2147483648 sampai 2147483647.
BIGINT	8	Bentuk terbesar dalam menangani data Numerik, mampu menangani data mulai dari -9223372036854775808 sampai 9223372036854775807.

### **TIPE DATA UNTUK PENANGGALAN DAN WAKTU**

Untuk menangani data-data yang berkaitan dengan waktu dan penanggalan, Anda dapat melihat beberapa tipe pada tabel berikut:

Type	Keterangan
DATETIME	Bentuk ini dapat menyimpan dua buah bentuk type data sekaligus, yaitu penanggalan dan waktu. Bentuk yang dapat diciptakan oleh DATETIME adalah '0000-00-00 00:00:00'. '0000-00-00' merupakan bentuk penanggalan yang dimulai dari tahun, bulan, dan tanggal. Sedangkan 00:00:00 adalah tempat menyimpan waktu atau jam. Misalnya: '2004-06-22 08:15:45'.
DATE	Bentuk ini digunakan untuk menyimpan data penanggalan saja, dengan bentuk penulisan '0000-00-00'. Penggunaan ini mirip seperti pada DATETIME, hanya saja yang ditampilkan hanya penanggalan saja. Misalnya, '2004-06-22'.
TIMESTAMP	Bentuk penanggalan dengan TIMESTAMP ditampilkan berjajar tanpa ada pembatasnya, dengan bentuk penulisan 0000000000000000 .
TIME	Bentuk TIME hanya digunakan untuk menyimpan data berbentuk jam. Yaitu dimulai dari tahun yang dibaca dari dua karakter terakhir dan selanjutnya diikuti bulan dan tanggal .bentuk penulisannya adalah '00:00:00' . pada prototype pewaktuan tersebut dapat dijabarkan bahwa 00 perma adalah jam yang diikuti menit dan detik. Contoh 08:35:55 .
YEAR	Bentuk yang paling sederhana adalah YEAR yang hanya menyimpan data berupa tahun saja. Ditulis secara lengkap 4 digit, misalnya: 2004 atau 1999 .

### **TIPE DATA STRING**

Dengan menyesuaikan banyaknya data, MySQL telah membagi datanya menjadi beberapa tipe, sehingga penggunaannya dapat disesuaikan. Perhatikan tabel berikut:

Type	Bytes	Keterangan
TINYTEXT		Type ini merupakan bentuk terkecil dari data String, yang mampu manangani data sampai dengan $2^{8-1}$ data .
TINYBLOB	255	Bentuk TINYTEXT adalah bentuk yang sama dengan TINYBLOB , yaitu mampu menangani data sampai dengan $2^{8-1}$ data .
TEXT	255	Bentuk TEXT salah satu bentuk type String yang mampu menangani data sampai dengan berukuran $2^{16-1}$ (64K-1) data.
BLOB	65535	Memiliki kemampuan sama dengan TEXT, yaitu sampai dengan $2^{16-1}$ (64K-1) data.
MEDIUMTEXT	65535	Dapat menyimpan data dengan ukuran cukup
	16777215	

MEDIUMBLOB	16777215	besar, sampai dengan $2^{24}-1$ (16M-1) data . Bentuk MEDIUMTEXT dapat Anda ganti dengan bentuk data MEDIUMBLOB, yang mampu menyimpan data sampai dengan $2^{24}-1$ (16M-1) data.
LONGBLOB	4294967295	Type data LONGBLOB adalah bentuk Type data yang paling besar dalam menangani data. Data yang disimpan sampai dengan berukuran Giga Byte. Type ini memiliki batasan penyimpanan sampai dengan $2^{32}-1$ (4G-1) data.

#### Daftar Tipe Data String yang Sering Digunakan

Type	Keterangan
VARCHAR	Bentuk ini dapat menyimpan data sampai dengan 225 karakter. Anda dapat menggunakan type ini apabila data yang dimasukan tidak lebih dari batasan tersebut.
CHAR	Bentuk CHAR hamper sama dengan VARCHAR, mampu menangani data sampai dengan 225 karakter. Namun, kedua type tersebut sangat signifikan dalam menyimpan data. Misalnya, Anda membuat kolom dengan Type VARCHAR(25). Meskipun Anda memasukan data kurang dari 25 digit, tetap dibaca sebanyak digit yang dimasukkan. Namun, jika Anda menggunakan type data CHAR(4), meskipun Anda memasukkan digit kurang dari 4, akan tetap dibaca 4 digit.
ENUM	Digunakan untuk validasi. Type data seperti ini, biasanya, kolom ditentukan terlebih dahulu. Misalnya, pada pembuatan kolom yang isinya mengenai golongan darah A, B, AB, dan O, bentuk penulisannya adalah ENUM('A','B','AB','O'). Jika memasukkan data tidak sesuai criteria, akan terjadi kesalahan atau tidak bisa dibaca (kosong).
SET	Type data SET sebenarnya memiliki fungsi yang sama dengan type ENUM, yaitu dengan mendeklarasikan anggota dari isi kolom yang mungkin akan menjadi anggotanya.

BLOB(Binary Large Object) merupakan tipe data yang biasa digunakan untuk menyimpan data berbentuk biner. Tipe data seperti LONGBLOB dapat digunakan untuk menyimpan gambar.

Berikut beberapa contoh kurang tepatnya pemilihan tipe data: 1) memilih CHAR(8) atau VARCHAR(10) dan bukannya DATE untuk menyimpan tanggal; kerugiannya, lebih boros tempat dan tidak bisa memanfaatkan fungsi-fungsi khusus tanggal; 2) memilih CHAR(3) atau CHAR(6) ketimbang TINYINT UNSIGNED untuk menyimpan data boolean ("YES" dan "NO"; atau "TRUE" dan "FALSE"; padahal jauh lebih irit dinyatakan dengan 1 dan 0 yang hanya menempati 1 byte); 3) memilih FLOAT atau DOUBLE dan bukannya DECIMAL

untuk menyimpan jumlah uang; kerugiannya, FLOAT dan DOUBLE adalah berbasis biner dan seringkali tidak eksak dalam menyimpan pecahan desimal.

Yang sering terjadi pada programer biasanya hanya mengenal single/double floating point number yang tersedia di bahasa pemrograman. Padahal database umumnya menyediakan angka pecahan berbasis desimal yang bisa digunakan menyimpan pecahan desimal.

## MEMBUAT TABEL

Pembuatan tabel berdasarkan desain yang sudah dibuat sebelumnya, tabel yang pertama dibuat adalah tabel **tblkelompok** dengan desain sebagai berikut;

### MEMBUAT TABEL **tblkelompok**

**tblkelompok**

	Field Name	Data Type
1	idkelompok	AutoNumber
2	kelompok	Text

Pastikan sebelum membuat tabel database sudah dipilih dengan benar; gunakan perintah berikut untuk pembuatan tabel. Pada desain yang dibuat menggunakan MS Access tipe data yang digunakan adalah **text** maka pada MySQL menggunakan **VARCHAR**

```
CREATE TABLE nama_tabel (nama_kolom TIPE_DATA, nama_kolom TIPE  
DATA);
```

```
MariaDB [dbtoko]> create table tblkelompok  
-> (  
-> idkelompok INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
-> kelompok VARCHAR(100)  
-> );  
Query OK, 0 rows affected (0.04 sec)
```

Untuk berpindah baris gunakan ENTER.

## MENAMPILKAN TABEL

Untuk menampilkan tabel yang sudah dibuat, gunakan perintah berikut;

**SHOW TABLES;**

```
MariaDB [dbtoko]> show tables;
+-----+
| Tables_in_dbtoko |
+-----+
| tblkelompok      |
+-----+
1 row in set (0.00 sec)
```

## MENAMPILKAN STRUKTUR TABEL

Untuk melihat struktur tabel gunakan perintah sebagai berikut;

**DESCRIBE nama\_tabel;**

```
MariaDB [dbtoko]> describe tblkelompok;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra        |
+-----+-----+-----+-----+-----+-----+
| idkelompok | int(11)    | NO   | PRI | NULL    | auto_increment |
| kelompok   | varchar(100) | YES  |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

### Membuat Tabel tblbarang

**tblbarang**

	Field Name	Data Type
!	idbarang	AutoNumber
!	idkelompok	Number
!	barang	Text
!	stok	Number
!	hargabeli	Currency
!	hargajual	Currency

Untuk membuat tabel tblbarang gunakan perintah sebagai berikut;

```
MariaDB [dbtoko]> CREATE TABLE tblbarang
-> (
-> idbarang INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
-> idkelompok INT,
-> barang VARCHAR(200),
-> stok FLOAT,
-> hargabeli FLOAT
-> );
Query OK, 0 rows affected (0.12 sec)
```

Karena pada MySQL tidak ada tipe data Currency maka kita bisa menggunakan tipe data **FLOAT**

Periksa apakah struktur yang dibuat sudah benar

```
MariaDB [dbtoko]> describe tblbarang;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO  | PRI | NULL    | auto_increment |
| idkelompok | int(11) | YES |     | NULL    |             |
| barang | varchar(200) | YES |     | NULL    |             |
| stok | float  | YES |     | NULL    |             |
| hargabeli | float  | YES |     | NULL    |             |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)
```

### Menampilkan Tabel

Tabel yang sudah dibuat pada database bisa dilihat dengan menggunakan perintah berikut;

```
MariaDB [dbtoko]> show tables;
+-----+
| Tables_in_dbtoko |
+-----+
| tblbarang        |
| tblkelompok      |
+-----+
2 rows in set (0.00 sec)
```

## MENAMBAH KOLOM

Jika dilihat pada desain terdapat kekurangan kolom pada tabel yang dibuat, kita bisa menambahkan kolom dengan perintah sebagai berikut;

```
ALTER TABLE nama_tabel ADD nama_kolom TIPE DATA;
```

```
MariaDB [dbtoko]> alter table tblbarang add hargajual float;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Setelah penambahan kolom periksa tblbarang dengan perintah sebagai berikut;

```
MariaDB [dbtoko]> describe tblbarang;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO   | PRI | NULL    | auto_increment |
| idkelompok | int(11) | YES  |     | NULL    |               |
| barang | varchar(200) | YES  |     | NULL    |               |
| stok | float   | YES  |     | NULL    |               |
| hargabeli | float   | YES  |     | NULL    |               |
| hargajual | float   | YES  |     | NULL    |               |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

## MERUBAH TIPE DATA

Pada tblbarang terdapat tipe data FLOAT pada kolom stok, kolom stok tipe data akan diganti menjadi INT. Gunakan perintah sebagai berikut;

```
ALTER TABLE nama_tabel MODIFY nama_kolom TIPE DATA;
```

```
MariaDB [dbtoko]> alter table tblbarang modify stok INT;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa kembali tipe data yang digunakan menggunakan perintah berikut;

```
MariaDB [dbtoko]> describe tblbarang;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO  | PRI | NULL    | auto_increment |
| idkelompok | int(11) | YES |     | NULL    |                |
| barang | varchar(200) | YES |     | NULL    |                |
| stok | int(11) | YES |     | NULL    |                |
| hargabeli | float  | YES |     | NULL    |                |
| hargajual | float  | YES |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

## MEMBERI NILAI DEFAULT PADA KOLOM

Pada kolom [**stok**] `tblbarang` akan diberi nilai DEFAULT [0] atau nilai awal [0]. Gunakan perintah sebagai berikut;

```
ALTER TABLE nama_tabel ALTER COLUMN nama_kolom SET DEFAULT
ISI_NILAI_DEFAULT_JIKA_VARCHAR_BERI_TANDA_PETIK;
```

```
MariaDB [dbtoko]> ALTER TABLE tblbarang ALTER COLUMN stok SET DEFAULT 0;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa dengan menggunakan perintah berikut;

```
MariaDB [dbtoko]> describe tblbarang;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO  | PRI | NULL    | auto_increment |
| idkelompok | int(11) | YES |     | NULL    |                |
| barang | varchar(200) | YES |     | NULL    |                |
| stok | int(11) | YES |     | 0       |                |
| hargabeli | float  | YES |     | NULL    |                |
| hargajual | float  | YES |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

## MENGHAPUS KOLOM

Pada latihan untuk menghapus kolom, tambahkan kolom baru terlebih dahulu dengan menggunakan perintah sebagai berikut;

```
MariaDB [dbtoko]> ALTER TABLE tblbarang ADD stokminimal INT;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa struktur tabel yang sudah dibuat;

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra
+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO   | PRI | NULL    | auto_increment
| idkelompok | int(11) | YES  |     | NULL    |
| barang | varchar(200) | YES  |     | NULL    |
| stok | int(11) | YES  |     | 0       |
| hargabeli | float  | YES  |     | NULL    |
| hargajual | float  | YES  |     | NULL    |
| stokminimal | int(11) | YES  |     | NULL    |
+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

Pada struktur diatas kolom stokminimal akan dihapus. Untuk menghapus gunakan perintah berikut;

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

```
MariaDB [dbtoko]> ALTER TABLE tblbarang DROP COLUMN stokminimal;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa struktur tabel dengan perintah berikut;

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra
+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO   | PRI | NULL    | auto_increment
| idkelompok | int(11) | YES  |     | NULL    |
| barang | varchar(200) | YES  |     | NULL    |
| stok | int(11) | YES  |     | 0       |
| hargabeli | float  | YES  |     | NULL    |
| hargajual | float  | YES  |     | NULL    |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

## MERUBAH NAMA KOLOM

Untuk merubah nama kolom jika terdapat kesalahan bisa menggunakan perintah sebagai berikut;

```
ALTER TABLE nama_tabel CHANGE kolom_lama kolom_baru TIPE DATA;
```

```
MariaDB [dbtoko]> ALTER TABLE tblbarang CHANGE stok stokbarang INT;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa struktur tabelnya;

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO  | PRI | NULL    | auto_increment |
| idkelompok | int(11) | YES |     | NULL    |               |
| barang | varchar(200) | YES |     | NULL    |               |
| stokbarang | int(11) | YES |     | NULL    |               |
| hargabeli | float   | YES |     | NULL    |               |
| hargajual | float   | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

## MEMBUAT TABEL tblpelanggan

tblpelanggan

tblpelanggan		
	Field Name	Data Type
idpelanggan		AutoNumber
nama		Text
alamat		Text

```
MariaDB [dbtoko]> CREATE TABLE tblpelanggan
-> (
-> idpelanggan INT,
-> nama VARCHAR(200),
-> alamat VARCHAR(255)
-> );
Query OK, 0 rows affected (0.05 sec)
```

Pada pembuatan tabel diatas belum ada PRIMARY KEY. Untuk menambahkan PRIMARY KEY gunakan perintah ubah tipe data.

## MENAMBAHKAN PRIMARY KEY

Perbaikan pada pembuatan tabel diatas dengan merubah tipe data pada kolom dan menambahkan PRIMARY KEY;

```
MariaDB [dbtoko]> ALTER TABLE tblpelanggan
    -> MODIFY idpelanggan INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa struktur tabel dengan perintah berikut;

```
MariaDB [dbtoko]> DESCRIBE tblpelanggan;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+
| idpelanggan | int(11) | NO   | PRI   | NULL    | auto_increment |
| nama        | varchar(200) | YES  |        | NULL    |              |
| alamat      | varchar(255) | YES  |        | NULL    |              |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

## MENAMPILKAN ENGINE YANG DIGUNAKAN

Pada pembuatan database terdapat 2 engine atau mesin yang digunakan yaitu **MYISAM** dan **InnoDB**. Untuk proses pembuatan database relasional harus menggunakan InnoDB. **MYISAM** tidak bisa menerima penggunaan FOREIGN KEY.

```
SHOW CREATE TABLE nama_tabel;
```

```
MariaDB [dbtoko]> SHOW CREATE TABLE tblpelanggan;
```

Hasil yang ditampilkan adalah;

```
| tblpelanggan | CREATE TABLE `tblpelanggan` (
  `idpelanggan` int(11) NOT NULL AUTO_INCREMENT,
  `nama` varchar(200) DEFAULT NULL,
  `alamat` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`idpelanggan`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

## MEMBUAT TABEL tblorder

tblorder

	Field Name	Data Type
!	idorder	AutoNumber
	idpelanggan	Number
	faktur	Text
	tanggalorder	Date/Time
	total	Currency
	bayar	Currency
	kembali	Currency

Perintah untuk membuat tblorder;

```
MariaDB [dbtoko]> CREATE TABLE tblorder
-> (
-> idorder INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
-> idpelanggan INT,
-> tanggalorder DATE,
-> total FLOAT,
-> bayar FLOAT,
-> kembali FLOAT
-> );
Query OK, 0 rows affected (0.04 sec)
```

Struktur tabel

```
MariaDB [dbtoko]> DESCRIBE tblorder;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idorder | int(11) | NO | PRI | NULL | auto_increment |
| idpelanggan | int(11) | YES | | NULL | |
| tanggalorder | date | YES | | NULL | |
| total | float | YES | | NULL | |
| bayar | float | YES | | NULL | |
| kembali | float | YES | | NULL | |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

Perhatikan pada desain pembuatan tabel diatas terdapat kekurangan satu kolom yaitu kolom [**faktur**] yang letaknya dibawah kolom [**idpelanggan**].

## MENAMBAH KOLOM SETELAH KOLOM

```
ALTER TABLE nama_tabel ADD nama_kolom TIPE DATA AFTER  
nama_kolom_sebelumnya;
```

```
MariaDB [dbtoko]> ALTER TABLE tblorder ADD faktur VARCHAR(50) AFTER idpelanggan;  
Query OK, 0 rows affected (0.09 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

Hasilnya adalah;

```
MariaDB [dbtoko]> DESCRIBE tblorder;  
+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra  
+-----+-----+-----+-----+  
| idorder | int(11) | NO | PRI | NULL | auto_increment  
| idpelanggan | int(11) | YES | | NULL |  
| faktur | varchar(50) | YES | | NULL |  
| tanggalorder | date | YES | | NULL |  
| total | float | YES | | NULL |  
| bayar | float | YES | | NULL |  
| kembali | float | YES | | NULL |  
+-----+-----+-----+-----+  
7 rows in set (0.01 sec)
```

## MEMBUAT TABEL tblorderdetail

tblorderdetail

tblorderdetail		
	Field Name	Data Type
1	idorderdetail	AutoNumber
2	idorder	Number
3	idbarang	Number
4	jumlah	Number
5	hargajual	Currency

Perintah membuat tblorderdetail

```
MariaDB [dbtoko]> CREATE TABLE tblorderdetail  
-> (  
-> idorderdetail INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
-> idorder INT,  
-> idbarang INT,  
-> jumlah INT,  
-> hargajual FLOAT  
-> );  
Query OK, 0 rows affected (0.04 sec)
```

Struktur tabel yang sudah dibuat;

```
MariaDB [dbtoko]> DESCRIBE tblorderdetail;
+-----+-----+-----+-----+-----+-----+
| Field      | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idorderdetail | int(11) | NO   | PRI  | NULL    | auto_increment |
| idorder     | int(11) | YES  |      | NULL    |               |
| idbarang    | int(11) | YES  |      | NULL    |               |
| jumlah      | int(11) | YES  |      | NULL    |               |
| hargajual   | float   | YES  |      | NULL    |               |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

## MEMBUAT INDEX

Index digunakan untuk mempercepat proses pencarian data terutama pada data yang jumlahnya diatas ribuan. Pada tabel diatas yang biasa digunakan untuk pencarian data adalah nama barang dan nama pelanggan.

```
CREATE INDEX nama_index ON nama_tabel (kolom);
```

```
MariaDB [dbtoko]> CREATE INDEX pelangganindex ON tblpelanggan (nama);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Buat index untuk tabel tblbarang dengan perintah sebagai berikut;

```
MariaDB [dbtoko]> CREATE INDEX barangindex ON tblbarang (barang);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

## MENAMPILKAN INDEX

Index yang sudah dibuat bisa ditampilkan dengan perintah;

```
SHOW INDEX FROM nama_tabel;
```

```
MariaDB [dbtoko]> SHOW INDEX FROM tblbarang;
```

Hasilnya adalah;

```
+-----+-----+-----+-----+
| Table | Non_unique | Key_name   | Seq_in_index | Column_name |
| Index_type | Comment | Index_comment |             |
+-----+-----+-----+-----+
| tblbarang |          0 | PRIMARY    |             1 | idbarang    |
| BTREE      |          |             |             1 | barang       |
| tblbarang |          1 | barangindex |             1 | barang       |
| BTREE      |          |             |             1 | barang       |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

## MENGHAPUS INDEX

Jika INDEX tidak digunakan bisa dihapus dengan perintah;

```
DROP INDEX nama_index ON nama_tabel;
```

```
MariaDB [dbtoko]> DROP INDEX pelangganindex ON tbpelanggan;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

# DML

***Data Manipulation Language***

**Insert, Delete, Update, Select, View, Trigger, Procedure, & Function**

## INSERT SEMUA KOLOM

Insert semua kolom digunakan untuk memasukan data pada semua kolom tabel. Sebelum melakukan INSERT data periksa dulu tabel yang akan digunakan.

```
MariaDB [dbtoko]> SHOW TABLES;
+-----+
| Tables_in_dbtoko |
+-----+
| tblbarang          |
| tblkelompok        |
| tblorder           |
| tblorderdetail     |
| tblpelanggan       |
+-----+
5 rows in set (0.01 sec)
```

Sebagai latihan akan menggunakan tblkelompok, lihat dulu kolom tblkelompok

```
MariaDB [dbtoko]> DESCRIBE tblkelompok;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra   |
+-----+-----+-----+-----+-----+-----+
| idkelompok | int(11)   | NO   | PRI | NULL    | auto_increment |
| kelompok   | varchar(100) | YES  |     | NULL    |           |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

Pada tblkelompok terdapat 2 kolom yang akan di INSERT datanya, yaitu [**idkelompok**], [**kelompok**].

```
INSERT INTO nama_tabel VALUES (kolom, kolom, semua_kolom);
```

```
MariaDB [dbtoko]> INSERT INTO tblkelompok VALUES('','Gula');
Query OK, 1 row affected, 1 warning (0.01 sec)
```

Untuk memeriksa apakah data yang di INSERT kan berhasil gunakan perintah;

```
SELECT * FROM nama_tabel;
```

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|         1   | Gula     |
+-----+-----+
1 row in set (0.00 sec)
```

## INSERT SEBAGIAN KOLOM

INSERT sebagian kolom digunakan untuk melakukan INSERT HANYA PADA kolom yang disebutkan.

```
INSERT INTO nama_tabel (kolom_yang_disebut) VALUES  
(isi_untuk_semua_kolom_yang_disebut);
```

```
MariaDB [dbtoko]> INSERT INTO tblkelompok (kelompok) VALUES('Beras');  
Query OK, 1 row affected (0.01 sec)
```

Sebagai latihan lakukan INSERT data sehingga diperoleh data sebagai berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;  
+-----+-----+  
| idkelompok | kelompok |  
+-----+-----+  
| 1 | Gula |  
| 2 | Beras |  
| 3 | Tepung |  
| 4 | Minyak |  
| 5 | Jajan |  
+-----+-----+  
5 rows in set (0.00 sec)
```

## DELETE SEBAGIAN RECORD (BARIS DATA)

Untuk menghapus sebagian baris gunakan perintah;

```
DELETE FROM nama_tabel WHERE baris_yang_akan_dihapus;
```

```
MariaDB [dbtoko]> DELETE FROM tblkelompok WHERE idkelompok=5;  
Query OK, 1 row affected (0.01 sec)
```

Baris Data (**Record**) setelah ada yang dihapus.

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;  
+-----+-----+  
| idkelompok | kelompok |  
+-----+-----+  
| 1 | Gula |  
| 2 | Beras |  
| 3 | Tepung |  
| 4 | Minyak |  
+-----+-----+  
4 rows in set (0.00 sec)
```

## **DELETE SEMUA RECORD**

Untuk menghapus semua baris data bisa dilakukan dengan perintah;

```
DELETE FROM nama_tabel;
```

```
MariaDB [dbtoko]> DELETE FROM tblkelompok;
Query OK, 4 rows affected (0.01 sec)
```

Periksa tabel yang sudah dihapus menggunakan perintah;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
Empty set (0.00 sec)
```

Karena data sudah dihapus maka yang tampil adalah KOSONG.

## **UPDATE SEBAGIAN RECORD**

Karena semua data sudah dihapus semua pada materi delete, isi dulu data pada tblkelompok sebagai latihan dengan data sebagai berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Jajan   |
|       7 | Minyak  |
|       8 | Tepung  |
+-----+-----+
3 rows in set (0.00 sec)
```

Pada tabel diatas **UBAH** data ‘Jajan’ menjadi ‘Snek’ menggunakan perintah sebagai berikut;

```
UPDATE nama_tabel SET nama_kolom=isi_kolom WHERE
nama_kolom=baris_yang_dipilih_untuk_diubah;
```

```
MariaDB [dbtoko]> UPDATE tblkelompok SET kelompok='Snek' WHERE idkelompok=6;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Periksa data yang sudah diubah

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Snek      |
|       7 | Minyak    |
|       8 | Tepung    |
+-----+-----+
3 rows in set (0.00 sec)
```

## UPDATE SEMUA RECORD

Untuk mengubah semua data bisa dilakukan dengan perintah sebagai berikut;

```
UPDATE nama_tabel SET nama_kolom=isi_kolom;
```

```
MariaDB [dbtoko]> UPDATE tblkelompok SET kelompok='Makanan';
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

Periksa data yang sudah diubah dengan perintah berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Makanan   |
|       7 | Makanan   |
|       8 | Makanan   |
+-----+-----+
3 rows in set (0.00 sec)
```

Lakukan UPDATE lagi sebagai latihan sehingga hasilnya sebagai berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras     |
|       7 | Gula      |
|       8 | Tepung    |
+-----+-----+
3 rows in set (0.00 sec)
```

## TABEL MASTER DAN TABEL TRANSAKSI (DETAIL)

Lihat struktur tblkelompok dan tblbarang;

### tblkelompok

```
MariaDB [dbtoko]> DESCRIBE tblkelompok;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra
+-----+-----+-----+-----+-----+
| idkelompok | int(11) | NO   | PRI   | NULL    | auto_increment |
| kelompok    | varchar(100)| YES  |        | NULL    |
+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

### tblbarang

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra
+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO   | PRI   | NULL    | auto_increment |
| idkelompok | int(11) | YES  |        | NULL    |
| barang    | varchar(200)| YES  | MUL   | NULL    |
| stokbarang | int(11) | YES  |        | NULL    |
| hargabeli | float   | YES  |        | NULL    |
| hargajual  | float   | YES  |        | NULL    |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

## TABEL MASTER

Tabel Master adalah tabel yang **MEMASUKI** tabel lain. Pada contoh diatas. Terdapat kolom [**idkelompok**] dari tabel [**tblkelompok**] yang **MEMASUKI** tabel [**tblbarang**]. Jadi tabel [**tblkelompok**] menjadi **TABEL MASTER**.

## TABEL TRANSAKSI

Tabel Transaksi adalah tabel yang **DIMASUKI** tabel lain. Pada contoh diatas tabel [**tblbarang**] **DIMASUKI** kolom [**idkelompok**] yang berasal dari tabel [**tblkelompok**]. Jadi yang tabel [**tblbarang**] menjadi **TABEL TRANSAKSI**.

## PRIMARY KEY DAN FOREIGN KEY

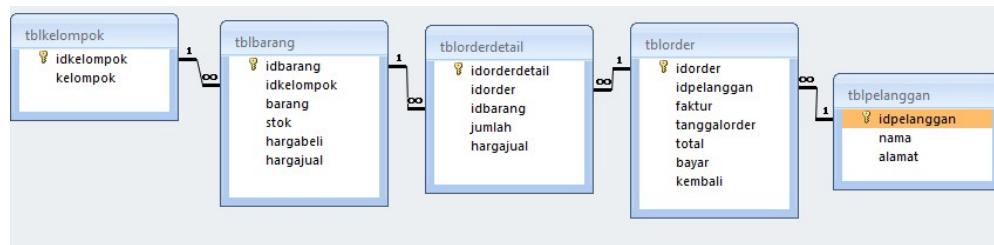
Pada tabel [**tblbarang**] terdapat kolom yang menjadi kunci atau KEY yang digunakan untuk membedakan baris data yang satu dengan dengan yang lain yaitu [**iDCLrang**]. Kolom yang berbeda antara baris data yang satu dengan yang lain disebut dengan **PRIMARY KEY**.

Pada **tblbarang** juga terdapat kolom yang masuk dari tabel lain yaitu [**tblkelompok**]. Kolom yang masuk ke **tblbarang** adalah [**idkelompok**]. Kolom yang masuk dari tabel lain disebut dengan **FOREIGN KEY**.

## **RELATIONAL DEPENDENCIES (HUBUNGAN KETERGANTUNGAN)**

DATABASE dibuat agar pengaturan data benar sesuai dengan yang diharapkan.

Relational Dependencies adalah pengaturan agar tabel yang satu terhubung dengan tabel yang lain. Sehingga data yang masuk ke satu tabel bergantung dari tabel yang lain dalam bentuk **BER-URUTAN**; Perhatikan desain database berikut;



Yang pertama kali bisa di isi adalah tabel [**tblkelompok**] dan [**tblpelanggan**]. Setelah tabel [**tblkelompok**] di isi maka tabel [**tblbarang**] baru bisa di isi. Jadi tabel [**tblbarang**] **TERGANTUNG (DEPENDENCIES)** dari tabel [**tblkelompok**]. Dibuat demikian agar setiap barang yang masuk pada tabel [**tblbarang**] mempunyai kelompok.

[**tblorder**] baru bisa di isi jika tabel [**tblpelanggan**] SUDAH di isi. Maka tabel [**tblorder**] **TERGANTUNG** pada tabel [**tblpelanggan**] sehingga hanya pelanggan yang sudah masuk pada tabel [**tblpelanggan**] saja yang bisa melakukan order.

[**tblorderdetail**] baru bisa di isi jika [**tblorder**] dan tabel [**tblbarang**] SUDAH di isi. Maka tabel [**tblorderdetail**] **TERGANTUNG** pada tabel [**tblorder**] dan [**tblpelanggan**]. Jadi hanya pelanggan yang sudah melakukan order saja yang bisa diambil barangnya. Ini semua disebut dengan **RELATIONAL DEPENDENCIES (HUBUNGAN KETERGANTUNGAN)**.

## **PROSES BISNIS (ALUR KERJA)**

Pada relasi tersebut terdapat alur kerja yang sering disebut dengan PROSES BISNIS.

Proses bisnis pada relasi tersebut yaitu:

Sebelum toko dibuka pemilik akan mengelompokan barang – barang yang akan dijual. Kelompok barang tersebut akan disimpan di tabel kelompok.

Contoh:

Toko akan menjual Beras, Gula, Tepung

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       8 | Tepung  |
+-----+-----+
3 rows in set (0.00 sec)
```

Kemudian pada kelompok tersebut akan dibuat DETAIL isinya. Perhatikan tabel dibawah;

<b>Kelompok</b>	<b>Isi kelompok</b>
Beras	Beras Rojo Lele
	Beras Cianjur
	Beras Medium
	Beras Mahal
Gula	Gula Putih
	Gula Merah
	Gula Batu
	Gula Aren
Tepung	Tepung Terigu
	Tepung Tapioka
	Tepung Kanji

Pada pembuatan database setiap kelompok akan diwakili oleh kolom PRIMARY KEY yang akan dimasukan ke tabel yang lain untuk menunjukan bahwa kolom tersebut adalah anggota dari tabel yang lain.

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO | PRI | NULL | auto_increment |
| idkelompok | int(11) | YES | | NULL | |
| barang | varchar(200) | YES | MUL | NULL | |
| stokbarang | int(11) | YES | | NULL | |
| hargabeli | float | YES | | NULL | |
| hargajual | float | YES | | NULL | |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

Pada tabel [**tblbarang**] diatas, [**idkelompok**] akan menunjukan bahwa kolom tersebut adalah isi dari tabel [**tblkelompok**]

## MEMBUAT RELASI ANTAR TABEL

Lihat dulu struktur tabel yang akan dibuat relasinya. Pada tblbarang terlihat struktur seperti pada gambar

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO   | PRI   | NULL    | auto_increment |
| idkelompok | int(11) | YES  |        | NULL    |             |
| barang | varchar(200) | YES  | MUL   | NULL    |             |
| stokbarang | int(11) | YES  |        | NULL    |             |
| hargabeli | float   | YES  |        | NULL    |             |
| hargajual | float   | YES  |        | NULL    |             |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

Perintah untuk membuat relasi adalah sebagai berikut;

```
ALTER TABLE nama_tabel_detail ADD CONSTRAINT
FK_foreign_key_pada_tabel_detail FOREIGN KEY (foreign_key) REFERENCES
nama_tabel_master (primary_key_tabel_master) ON UPDATE CASCADE ON
DELETE RESTRICT;
```

```
MariaDB [dbtoko]> ALTER TABLE tblbarang
    -> ADD CONSTRAINT FK_idkelompok FOREIGN KEY (idkelompok)
    -> REFERENCES tbkelompok (idkelompok) ON UPDATE CASCADE
    -> ON DELETE RESTRICT;
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

**ON UPDATE CASCADE** : jika kolom primary key pada tabel master maka kolom foreign key pada tabel transaksi akan berubah.

**ON DELETE RESTRICT** : Kolom primary key di tabel master **TIDAK BISA** dihapus jika kolom tersebut masih digunakan pada tabel transaksi. **ON DELETE CASCADE** jika baris pada kolom master dihapus maka semua baris data yang menggunakan isi tabel master pada tabel transaksi akan terhapus :

Untuk melihat hasil pembuatan relasi gunakan perintah berikut;

```
MariaDB [dbtoko]> SHOW CREATE TABLE tblbarang;
```

Tampilan jika relasi berhasil dibuat adalah;

```
| tblbarang | CREATE TABLE `tblbarang` (| `idbarang` int(11) NOT NULL AUTO_INCREMENT,| `idkelompok` int(11) DEFAULT NULL,| `barang` varchar(200) DEFAULT NULL,| `stokbarang` int(11) DEFAULT NULL,| `hargabeli` float DEFAULT NULL,| `hargajual` float DEFAULT NULL,| PRIMARY KEY (`idbarang`),| KEY `barangindex` (`barang`),| KEY `FK_idkelompok` (`idkelompok`),| CONSTRAINT `FK_idkelompok` FOREIGN KEY (`idkelompok`) REFERENCES| `tblkelompok` (`idkelompok`) ON UPDATE CASCADE,| CONSTRAINT `FK_tblbarang_tblkelompok` FOREIGN KEY (`idkelompok`)| REFERENCES `tblkelompok` (`idkelompok`) ON UPDATE CASCADE| ) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

## MENGUJI HASIL PEMBUATAN RELASI

Untuk menguji hasil pembuatan relasi lakukan INSERT data pada tabel master terlebih dahulu baru kemudian ke tabel transaksi.

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;+-----+-----+| idkelompok | kelompok |+-----+-----+| 6 | Beras | | 7 | Gula | | 8 | Tepung |+-----+-----+3 rows in set (0.00 sec)
```

Tampilkan struktur tabel untuk memudahkan proses insert data;

```
MariaDB [dbtoko]> DESCRIBE tblbarang;+-----+-----+-----+-----+-----+-----+| Field | Type | Null | Key | Default | Extra |+-----+-----+-----+-----+-----+-----+| idbarang | int(11) | NO | PRI | NULL | auto_increment | | idkelompok | int(11) | YES | MUL | NULL | | barang | varchar(200) | YES | MUL | NULL | | stokbarang | int(11) | YES | | NULL | | hargabeli | float | YES | | NULL | | hargajual | float | YES | | NULL |+-----+-----+-----+-----+-----+-----+6 rows in set (0.00 sec)
```

Isi tabel transaksi sesuai dengan idkelompok yang akan digunakan

Karena iDCLrang AUTO\_INCREMENT maka kolom bisa diganti dengan "" (petik kosong)

```
MariaDB [dbtoko]> INSERT INTO tblbarang VALUES ('','6','Beras Rojo Lele',100,10000,12000);
Query OK, 1 row affected, 1 warning (0.01 sec)
```

Lakukan insert data sehingga muncul tabel berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblbarang;
+-----+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+-----+
| 1 | 6 | Beras Rojo Lele | 100 | 10000 | 12000 |
| 2 | 6 | Beras Cianjur | 50 | 11000 | 14000 |
| 3 | 6 | Beras Medium | 70 | 8000 | 10000 |
| 4 | 6 | Beras Mahal | 30 | 23000 | 30000 |
| 5 | 7 | Gula Putih | 20 | 12000 | 14000 |
| 6 | 7 | Gula Merah | 10 | 3000 | 5000 |
| 7 | 7 | Gula Batu | 40 | 2000 | 3000 |
| 8 | 7 | Gula Aren | 60 | 7000 | 9000 |
| 10 | 8 | Tepung Terigu | 50 | 4000 | 6000 |
| 11 | 8 | Tepung Tapioka | 15 | 2000 | 3500 |
| 12 | 8 | Tepung Kanji | 25 | 3500 | 5000 |
+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

## PENGUJIAN INSERT

Insert pada tabel [tblbarang] **HANYA BISA** dilakukan menggunakan data yang tersedia pada tabel master. [**Idkelompok**] pada tabel master yang tersedia hanya [6,7,8]..

Perhatikan hasil pengujian berikut;

Pada pengujian INSERT idkelompok yang digunakan adalah 9 yang **TIDAK TERSEDIA** pada tabel master [**tblkelompok**].

```
MariaDB [dbtoko]> INSERT INTO tblbarang VALUES ('',9,'Tepung Sagu',20,4500,7000);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
(`dbtoko`.`tblbarang`, CONSTRAINT `FK_idkelompok` FOREIGN KEY (`idkelompok`) REFERENCES `tblkelompok` (`idkelompok`) ON UPDATE CASCADE)
MariaDB [dbtoko]>
```

## PENGUJIAN UPDATE

Coba lakukan update pada tabel master kemudian periksa di tabel transaksi.

Ubah idkelompok tepung dari 8 menjadi 9 dengan perintah berikut;

```
MariaDB [dbtoko]> UPDATE tblkelompok SET idkelompok=9 WHERE kelompok='Tepung';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

Periksa perubahan pada tblkelompok;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
+-----+-----+
3 rows in set (0.00 sec)
```

Periksa pada tblbarang;

```
MariaDB [dbtoko]> SELECT * FROM tblbarang;
+-----+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang      | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+-----+
|       1 |       6 | Beras Rojo Lele |      100 |     10000 |     12000 |
|       2 |       6 | Beras Cianjur   |       50 |     11000 |     14000 |
|       3 |       6 | Beras Medium    |       70 |      8000 |     10000 |
|       4 |       6 | Beras Mahal     |       30 |     23000 |     30000 |
|       5 |       7 | Gula Putih      |       20 |     12000 |     14000 |
|       6 |       7 | Gula Merah      |       10 |      3000 |      5000 |
|       7 |       7 | Gula Batu       |       40 |      2000 |      3000 |
|       8 |       7 | Gula Aren       |       60 |      7000 |      9000 |
|      10 |       9 | Tepung Terigu   |       50 |      4000 |      6000 |
|      11 |       9 | Tepung Tapioka  |       15 |      2000 |      3500 |
|      12 |       9 | Tepung Kanji    |       25 |      3500 |      5000 |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

Pada ON UPDATE CASCADE jika kolom tabel master berubah maka kolom tabel transaksi juga berubah

## PENGUJIAN DELETE

Pengujian delete dilakukan dengan menghapus tabel master.

```
MariaDB [dbtoko]> DELETE FROM tblkelompok WHERE idkelompok=9;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
(`dbtoko`.`tblbarang`, CONSTRAINT `FK_tblbarang_tblkelompok` FOREIGN KEY (`idkelompok`)
 REFERENCES `tblkelompok` (`idkelompok`) ON UPDATE CASCADE)
MariaDB [dbtoko]>
```

Karena [ **idkelompok** ] 9 sudah digunakan pada tabel transaksi maka data tersebut tidak bisa dihapus seuai dengan relasi yang dibuat yaitu **ON DELETE RESTRICT**. Data yang bisa dihapus hanya yang belum digunakan pada tabel transaksi. coba INSERT kan data baru pada tabel master;

```
MariaDB [dbtoko]> INSERT INTO tblkelompok VALUES (10,'Minyak');
Query OK, 1 row affected (0.01 sec)
```

Periksa data yang sudah dimasukan

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
|      10 | Minyak |
+-----+-----+
4 rows in set (0.00 sec)
```

Karena kolom [idkelompok] 10 belum digunakan pada tabel transaksi maka kolom tersebut bisa dihapus. Lakukan penghapusan menggunakan perintah berikut;

```
MariaDB [dbtoko]> DELETE FROM tblkelompok WHERE idkelompok=10;
Query OK, 1 row affected (0.01 sec)
```

Periksa tabel yang sudah dihapus;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
+-----+-----+
3 rows in set (0.00 sec)
```

## PEMBUATAN VIEW

View digunakan untuk menggabungkan tabel master dan tabel transaksi menjadi sebuah tabel baru agar lebih mudah dilihat dan digunakan.

```
CREATE VIEW nama_view AS SELECT tabel_transaksi.kolom_transaksi,
tabel_master.kolom_master FROM tabel_transaksi INNER JOIN tabel_master
ON tabel_transaksi.kolom_transaksi = tabel_master.kolom_master;
```

```
MariaDB [dbtoko]> CREATE VIEW view_barang AS SELECT
-> tblbarang.idbarang, tblbarang.barang,
-> tblbarang.stokbarang, tblbarang.hargabeli,
-> tblbarang.hargajual, tblbarang.idkelompok,
-> tblkelompok.kelompok FROM tblbarang
-> INNER JOIN tblkelompok ON tblbarang.idkelompok = tblkelompok.idkelompok;
Query OK, 0 rows affected (0.01 sec)
```

## MENAMPILKAN SEMUA VIEW YANG SUDAH DIBUAT

View yang sudah dibuat bisa dilihat dengan perintah;

```
SHOW FULL TABLES IN nama_database WHERE TABLE_TYPE LIKE 'VIEW';
```

```
MariaDB [dbtoko]> SHOW FULL TABLES IN dbtoko WHERE TABLE_TYPE LIKE 'VIEW';
+-----+-----+
| Tables_in_dbtoko | Table_type |
+-----+-----+
| view_barang      | VIEW       |
+-----+-----+
1 row in set (0.00 sec)
```

## MELIHAT ISI VIEW

Untuk melihat isi VIEW sama dengan melihat isi tabel

```
SELECT * FROM nama_view;
```

```
MariaDB [dbtoko]> SELECT * FROM view_barang;
+-----+-----+-----+-----+-----+-----+
| idbarang | barang      | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+-----+
| 1 | Beras Rojo Lele | 100 | 10000 | 12000 | 6 | Beras |
| 2 | Beras Cianjur | 50 | 11000 | 14000 | 6 | Beras |
| 3 | Beras Medium | 70 | 8000 | 10000 | 6 | Beras |
| 4 | Beras Mahal | 30 | 23000 | 30000 | 6 | Beras |
| 5 | Gula Putih | 20 | 12000 | 14000 | 7 | Gula |
| 6 | Gula Merah | 10 | 3000 | 5000 | 7 | Gula |
| 7 | Gula Batu | 40 | 2000 | 3000 | 7 | Gula |
| 8 | Gula Aren | 60 | 7000 | 9000 | 7 | Gula |
| 10 | Tepung Terigu | 50 | 4000 | 6000 | 9 | Tepung |
| 11 | Tepung Tapioka | 15 | 2000 | 3500 | 9 | Tepung |
| 12 | Tepung Kanji | 25 | 3500 | 5000 | 9 | Tepung |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

## MENGHAPUS VIEW

Jika VIEW yang sudah dibuat tidak digunakan lagi bisa dihapus dengan menggunakan perintah berikut;

```
DROP VIEW nama_view;
```

```
MariaDB [dbtoko]> DROP VIEW view_barang;
Query OK, 0 rows affected (0.00 sec)
```

Periksa daftar view dengan perintah berikut;

```
MariaDB [dbtoko]> SHOW FULL TABLES IN dbtoko WHERE TABLE_TYPE LIKE 'VIEW';
Empty set (0.00 sec)
```

## **SELECT SEMUA KOLOM (\*)**

Select semua adalah menampilkan semua kolom dari tabel atau VIEW

```
SELECT * FROM nama_tabel_atau_nama_view;
```

```
MariaDB [dbtoko]> SELECT * FROM view_barang;
+-----+-----+-----+-----+-----+-----+
| idbarang | barang | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+-----+
| 1 | Beras Rojo Lele | 100 | 10000 | 12000 | 6 | Beras |
| 2 | Beras Cianjur | 50 | 11000 | 14000 | 6 | Beras |
| 3 | Beras Medium | 70 | 8000 | 10000 | 6 | Beras |
| 4 | Beras Mahal | 30 | 23000 | 30000 | 6 | Beras |
| 5 | Gula Putih | 20 | 12000 | 14000 | 7 | Gula |
| 6 | Gula Merah | 10 | 3000 | 5000 | 7 | Gula |
| 7 | Gula Batu | 40 | 2000 | 3000 | 7 | Gula |
| 8 | Gula Aren | 60 | 7000 | 9000 | 7 | Gula |
| 10 | Tepung Terigu | 50 | 4000 | 6000 | 9 | Tepung |
| 11 | Tepung Tapioka | 15 | 2000 | 3500 | 9 | Tepung |
| 12 | Tepung Kanji | 25 | 3500 | 5000 | 9 | Tepung |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

## **SELECT SEBAGIAN KOLOM**

Select sebagian kolom adalah menampilkan hanya pada kolom yang dipilih

```
SELECT nama_kolom, nama_kolom FROM nama_tabel_atau_nama_view;
```

```
MariaDB [dbtoko]> SELECT barang,stokbarang,kelompok FROM view_barang;
+-----+-----+-----+
| barang | stokbarang | kelompok |
+-----+-----+-----+
| Beras Rojo Lele | 100 | Beras |
| Beras Cianjur | 50 | Beras |
| Beras Medium | 70 | Beras |
| Beras Mahal | 30 | Beras |
| Gula Putih | 20 | Gula |
| Gula Merah | 10 | Gula |
| Gula Batu | 40 | Gula |
| Gula Aren | 60 | Gula |
| Tepung Terigu | 50 | Tepung |
| Tepung Tapioka | 15 | Tepung |
| Tepung Kanji | 25 | Tepung |
+-----+-----+-----+
11 rows in set (0.00 sec)
```

## SELECT ORDER

Jika VIEW pada latihan diatas sudah dihapus, silahkan buat ulang untuk latihan materi selanjutnya;

```
MariaDB [dbtoko]> CREATE VIEW view_barang AS SELECT
    -> tblbarang.idbarang, tblbarang.barang,
    -> tblbarang.stokbarang, tblbarang.hargabeli,
    -> tblbarang.hargajual, tblbarang.idkelompok,
    -> tblkelompok.kelompok FROM tblbarang
    -> INNER JOIN tblkelompok ON tblbarang.idkelompok = tblkelompok.idkelompok;
Query OK, 0 rows affected (0.01 sec)
```

SELECT ORDER adalah cara untuk menampilkan data dalam urutan naik atau turun

Jika NAIK menggunakan ASC

Jika TURUN menggunakan DESC

```
SELECT * FROM nama_tabel_atau_view ORDER BY nama_kolom ASC;
```

```
MariaDB [dbtoko]> SELECT * FROM view_barang ORDER BY hargabeli ASC;
```

Hasil select ORDER BY nama\_kolom ASC

```
MariaDB [dbtoko]> SELECT * FROM view_barang ORDER BY hargabeli ASC;
+-----+-----+-----+-----+-----+-----+
| idbarang | barang | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+-----+
| 11 | Tepung Tapioka | 15 | 2000 | 3500 | 9 | Tepung |
| 7 | Gula Batu | 40 | 2000 | 3000 | 7 | Gula |
| 6 | Gula Merah | 10 | 3000 | 5000 | 7 | Gula |
| 12 | Tepung Kanji | 25 | 3500 | 5000 | 9 | Tepung |
| 10 | Tepung Terigu | 50 | 4000 | 6000 | 9 | Tepung |
| 8 | Gula Aren | 60 | 7000 | 9000 | 7 | Gula |
| 3 | Beras Medium | 70 | 8000 | 10000 | 6 | Beras |
| 1 | Beras Rojo Lele | 100 | 10000 | 12000 | 6 | Beras |
| 2 | Beras Cianjur | 50 | 11000 | 14000 | 6 | Beras |
| 5 | Gula Putih | 20 | 12000 | 14000 | 7 | Gula |
| 4 | Beras Mahal | 30 | 23000 | 30000 | 6 | Beras |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

Hasil select ORDER BY nama\_kolom DESC

```
MariaDB [dbtoko]> SELECT * FROM view_barang ORDER BY hargabeli DESC;
+-----+-----+-----+-----+-----+-----+
| idbarang | barang | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+-----+
| 4 | Beras Mahal | 30 | 23000 | 30000 | 6 | Beras |
| 5 | Gula Putih | 20 | 12000 | 14000 | 7 | Gula |
| 2 | Beras Cianjur | 50 | 11000 | 14000 | 6 | Beras |
| 1 | Beras Rojo Lele | 100 | 10000 | 12000 | 6 | Beras |
| 3 | Beras Medium | 70 | 8000 | 10000 | 6 | Beras |
| 8 | Gula Aren | 60 | 7000 | 9000 | 7 | Gula |
| 10 | Tepung Terigu | 50 | 4000 | 6000 | 9 | Tepung |
| 12 | Tepung Kanji | 25 | 3500 | 5000 | 9 | Tepung |
| 6 | Gula Merah | 10 | 3000 | 5000 | 7 | Gula |
| 11 | Tepung Tapioka | 15 | 2000 | 3500 | 9 | Tepung |
| 7 | Gula Batu | 40 | 2000 | 3000 | 7 | Gula |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

## SELECT GROUP

Perintah **SELECT GROUP** digunakan untuk mengelompokkan data sesuai dengan kelompok dari kolom yang dipilih.

```
SELECT * FROM nama_tabel_atau_view GROUP BY nama_kolom;
```

```
MariaDB [dbtoko]> SELECT * FROM view_barang GROUP BY barang;
+-----+-----+-----+-----+-----+-----+
| idbarang | barang      | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+-----+
| 2 | Beras Cianjur |      50 |   11000 |   14000 |       6 | Beras   |
| 4 | Beras Mahal   |      30 |   23000 |   30000 |       6 | Beras   |
| 3 | Beras Medium  |      70 |    8000 |   10000 |       6 | Beras   |
| 1 | Beras Rojo Lele|     100 |  10000 |  12000 |       6 | Beras   |
| 8 | Gula Aren     |      60 |    7000 |   9000  |       7 | Gula    |
| 7 | Gula Batu     |      40 |    2000 |   3000  |       7 | Gula    |
| 6 | Gula Merah    |      10 |    3000 |   5000  |       7 | Gula    |
| 5 | Gula Putih    |      20 |   12000 |  14000 |       7 | Gula    |
| 12 | Tepung Kanji  |      25 |   3500  |   5000  |       9 | Tepung  |
| 11 | Tepung Tapioka|      15 |   2000  |   3500  |       9 | Tepung  |
| 10 | Tepung Terigu  |      50 |   4000  |   6000  |       9 | Tepung  |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

## PENGUJIAN WHERE

Pengujian WHERE pengujian pada SQL yang bisa diletakan di belakang:

- SELECT
- DELETE
- UPDATE

Pengujian akan menjalankan SELECT, DELETE, UPDATE jika kondisi yang diuji terpenuhi

- Operator pembanding [=, >, <, <>, >=, <=, LIKE]
- Operator LOGIKA (AND, OR)
- SELECT

Contoh pengujian

```
SELECT * FROM nama_tabel_atau_view WHERE pengujian ORDER BY
nama_kolom;
```

```
MariaDB [dbtoko]> SELECT * FROM tblbarang WHERE hargajual > 5000 ORDER BY hargajual;
+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang      | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+
| 10 |         9 | Tepung Terigu |      50 |   4000 |   6000 |
| 8 |         7 | Gula Aren     |      60 |   7000 |  9000  |
| 3 |         6 | Beras Medium  |      70 |   8000 | 10000 |
| 1 |         6 | Beras Rojo Lele|     100 |  10000 | 12000 |
| 2 |         6 | Beras Cianjur |      50 |   11000 | 14000 |
| 5 |         7 | Gula Putih    |      20 |   12000 | 14000 |
| 4 |         6 | Beras Mahal   |      30 |   23000 | 30000 |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

```
MariaDB [dbtoko]> SELECT * FROM tblbarang
    -> WHERE hargajual > 5000
    -> AND hargabeli > 7000
    -> ORDER BY hargabeli;
+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang      | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+
|     3 |         6 | Beras Medium |       70 |     8000 |   10000 |
|     1 |         6 | Beras Rojo Lele |      100 |    10000 |   12000 |
|     2 |         6 | Beras Cianjur |       50 |    11000 |   14000 |
|     5 |         7 | Gula Putih |       20 |    12000 |   14000 |
|     4 |         6 | Beras Mahal |       30 |    23000 |   30000 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
MariaDB [dbtoko]> SELECT * FROM tblbarang
    -> WHERE hargajual > 5000
    -> OR hargabeli > 7000
    -> ORDER BY hargabeli;
+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang      | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+
|    10 |         9 | Tepung Terigu |       50 |     4000 |   6000 |
|     8 |         7 | Gula Aren |       60 |     7000 |   9000 |
|     3 |         6 | Beras Medium |       70 |     8000 | 10000 |
|     1 |         6 | Beras Rojo Lele |      100 |    10000 | 12000 |
|     2 |         6 | Beras Cianjur |       50 |    11000 | 14000 |
|     5 |         7 | Gula Putih |       20 |    12000 | 14000 |
|     4 |         6 | Beras Mahal |       30 |    23000 | 30000 |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

**SELECT \* FROM nama\_tabel\_atau\_view WHERE nama\_kolom LIKE '%\_apapun';**

% adalah sebutan untuk sembarang atau apapun

Contoh:

**%a artinya awalan sembarang yang penting akhirnya a**

```
MariaDB [dbtoko]> SELECT * FROM view_barang WHERE barang LIKE '%a';
+-----+-----+-----+-----+-----+
| idbarang | barang      | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+
|    11 | Tepung Tapioka |       15 |    2000 |    3500 |        9 | Tepung    |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

**b% artinya awalan b akhiran sembarang**

```
MariaDB [dbtoko]> SELECT * FROM view_barang WHERE barang LIKE 'b%';
+-----+-----+-----+-----+-----+
| idbarang | barang      | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+
|     1 | Beras Rojo Lele |      100 | 10000 | 12000 |        6 | Beras    |
|     2 | Beras Cianjur |       50 | 11000 | 14000 |        6 | Beras    |
|     3 | Beras Medium |       70 |  8000 | 10000 |        6 | Beras    |
|     4 | Beras Mahal |       30 | 23000 | 30000 |        6 | Beras    |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

**%d% artinya awalan sembarang dan akhiran sembarang yang tengahnya ada huruf d**

```
MariaDB [dbtoko]> SELECT * FROM view_barang WHERE barang LIKE '%d%';
+-----+-----+-----+-----+-----+
| idbarang | barang      | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+
|      3 | Beras Medium |         70 |     8000 |    10000 |          6 | Beras      |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

## SUBQUERY (SELECT IN SELECT)

Subquery adalah SELECT yang ada di dalam SELECT. Sebelum berlatih menggunakan subquery tambahkan data pada [tblkelompok].

```
MariaDB [dbtoko]> INSERT INTO tblkelompok VALUES ('','Minyak');
Query OK, 1 row affected, 1 warning (0.01 sec)

MariaDB [dbtoko]> INSERT INTO tblkelompok VALUES ('','Jajan');
Query OK, 1 row affected, 1 warning (0.01 sec)

MariaDB [dbtoko]> INSERT INTO tblkelompok VALUES ('','Roti Basah');
Query OK, 1 row affected, 1 warning (0.01 sec)
```

sehingga datanya akan tampil seperti berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok   |
+-----+-----+
|      6 | Beras      |
|      7 | Gula       |
|      9 | Tepung     |
|     10 | Minyak     |
|     11 | Jajan      |
|     12 | Roti Basah |
+-----+-----+
6 rows in set (0.00 sec)
```

### Periksa [view\_barang]

```
MariaDB [dbtoko]> SELECT * FROM view_barang;
+-----+-----+-----+-----+-----+-----+
| idbarang | barang | stokbarang | hargabeli | hargajual | idkelompok | kelompok |
+-----+-----+-----+-----+-----+-----+
| 1 | Beras Rojo Lele | 100 | 10000 | 12000 | 6 | Beras |
| 2 | Beras Cianjur | 50 | 11000 | 14000 | 6 | Beras |
| 3 | Beras Medium | 70 | 8000 | 10000 | 6 | Beras |
| 4 | Beras Mahal | 30 | 23000 | 30000 | 6 | Beras |
| 5 | Gula Putih | 20 | 12000 | 14000 | 7 | Gula |
| 6 | Gula Merah | 10 | 3000 | 5000 | 7 | Gula |
| 7 | Gula Batu | 40 | 2000 | 3000 | 7 | Gula |
| 8 | Gula Aren | 60 | 7000 | 9000 | 7 | Gula |
| 10 | Tepung Terigu | 50 | 4000 | 6000 | 9 | Tepung |
| 11 | Tepung Tapioka | 15 | 2000 | 3500 | 9 | Tepung |
| 12 | Tepung Kanji | 25 | 3500 | 5000 | 9 | Tepung |
+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

Pada [view\_barang] ada kelompok yang sudah menggunakan data dari tabel [tblkelompok]. Ada pertanyaan yang muncul dari data diatas adalah:

1. Ada berapa data dari tabel [tblkelompok] yang **SUDAH** digunakan pada VIEW?
2. Ada berapa data yang **BELUM** digunakan pada VIEW?

Untuk menjawab pertanyaan diatas bisa menggunakan **SUBQUERY**.

```
SELECT * FROM nama_tabel WHERE nama_kolom IN (SELECT nama_kolom FROM
nama_tabel)
```

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok
    -> WHERE idkelompok IN
    -> (SELECT idkelompok FROM view_barang);
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
| 6 | Beras |
| 7 | Gula |
| 9 | Tepung |
+-----+-----+
3 rows in set (0.00 sec)
```

```
SELECT * FROM nama_tabel WHERE nama_kolom NOT IN (SELECT nama_kolom FROM nama_tabel)
```

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok
    -> WHERE idkelompok NOT IN
    -> (SELECT idkelompok FROM view_barang);
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|      10 | Minyak   |
|      11 | Jajan    |
|      12 | Roti Basah|
+-----+-----+
3 rows in set (0.00 sec)
```

```
PENGGUNAAN TANDA * PADA SELECT IN HANYA BISA DILAKUKAN PADA SELECT YANG PERTAMA
```

## MEMBUAT RELASI ANTAR TABEL

pada desain database terdapat relasi antar tabel [**tblpelanggan**] dan tabel [**tblorder**]. Lengkapi proses pembuatan relasi pada tabel [**tblorder**].

### PROSES BISNIS

Pelanggan yang akan membeli akan dicatat di tabel order. Hanya pelanggan yang sudah terdaftar pada tabel pelanggan yang bisa melakukan order.



```
MariaDB [dbtoko]> ALTER TABLE tblorder
    -> ADD CONSTRAINT FK_idpelanggan FOREIGN KEY (idpelanggan)
    -> REFERENCES tblpelanggan (idpelanggan)
    -> ON UPDATE CASCADE ON DELETE RESTRICT;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa hasil pembuatan RELASI dengan perintah berikut;

```
MariaDB [dbtoko]> SHOW CREATE TABLE tblorder;
```

Hasil pembuatan relasi

```
tblorder | CREATE TABLE `tblorder` (
  `idorder` int(11) NOT NULL AUTO_INCREMENT,
  `idpelanggan` int(11) DEFAULT NULL,
  `faktur` varchar(50) DEFAULT NULL,
  `tanggalorder` date DEFAULT NULL,
  `total` float DEFAULT NULL,
  `bayar` float DEFAULT NULL,
  `kembali` float DEFAULT NULL,
  PRIMARY KEY (`idorder`),
  KEY `FK_idpelanggan` (`idpelanggan`),
  CONSTRAINT `FK_idpelanggan` FOREIGN KEY (`idpelanggan`) REFERENCES `tblpelanggan` (`idpelanggan`) ON UPDATE CASCADE
ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1 |
```

## RELASI 2 TABEL MASTER DAN 1 TABEL TRANSAKSI

Pada tabel [tblorderdetail] detail terdapat relasi antara tabel [tblorder] dan tabel [tblbarang]



Buat relasi antara tabel [tblorderdetail] dan tabel [tblorder]

```
MariaDB [dbtoko]> ALTER TABLE tblorderdetail
    -> ADD CONSTRAINT FK_idorder FOREIGN KEY (idorder)
    -> REFERENCES tblorder (idorder)
    -> ON UPDATE CASCADE ON DELETE RESTRICT;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Buat relasi antara tabel [tblorderdetail] dan tabel [tblbarang]

```
MariaDB [dbtoko]> ALTER TABLE tblorderdetail
    -> ADD CONSTRAINT FK_idbarang FOREIGN KEY (idbarang)
    -> REFERENCES tblbarang (idbarang)
    -> ON UPDATE CASCADE ON DELETE RESTRICT;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa hasil pembuatan relasi antara 2 tabel master [**tblorder**] [**tblbarang**] dan 1 tabel transaksi [**tblorderdetail**]

```
MariaDB [dbtoko]> SHOW CREATE TABLE tblorderdetail;
```

Hasil pembuatan relasi

```
tblorderdetail | CREATE TABLE `tblorderdetail` (
`idorderdetail` int(11) NOT NULL AUTO_INCREMENT,
`idorder` int(11) DEFAULT NULL,
`idbarang` int(11) DEFAULT NULL,
`jumlah` int(11) DEFAULT NULL,
`hargajual` float DEFAULT NULL,
PRIMARY KEY (`idorderdetail`),
KEY `FK_idorden` (`idorder`),
KEY `FK_idbarang` (`idbarang`),
CONSTRAINT `FK_idbarang` FOREIGN KEY (`idbarang`) REFERENCES `tblbarang` (`idbarang`) ON UPDATE CASCADE,
CONSTRAINT `FK_idorden` FOREIGN KEY (`idorder`) REFERENCES `tblorder` (`idorder`) ON UPDATE CASCADE
ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

## DUMMY DATA

Dummy data adalah data yang tidak sebenarnya. Data ini diperlukan jika data sebenarnya tidak bisa diperoleh. Dummy data biasa digunakan selama pengujian database.

Pada materi ini akan menggunakan dummy data pada nama pelanggan. Umumnya pelanggan mempunyai nama. Jika anda berbelanja pada supermarket atau minimarket maka anda tidak akan ditanya tentang nama. Anda hanya akan ditanya yang anda beli. Pertanyaannya bagaimana aplikasi mencatat nama anda?

Pada tabel [**tblpelanggan**] aplikasi akan diisi dengan dummy data dengan nama pelanggan sebagai berikut;

Tampilkan struktur tabel terlebih dahulu

```
MariaDB [dbtoko]> DESCRIBE tblpelanggan;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+
| idpelanggan | int(11)    | NO   | PRI | NULL    | auto_increment |
| nama        | varchar(200) | YES  |     | NULL    |                |
| alamat      | varchar(255) | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Isikan nama pelanggan dengan dummy data

```
MariaDB [dbtoko]> INSERT INTO tblpelanggan VALUES('','KOSONG','KOSONG');
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Periksa hasil INSERT datanya;

```
MariaDB [dbtoko]> SELECT * FROM tblpelanggan;
+-----+-----+-----+
| idpelanggan | nama    | alamat   |
+-----+-----+-----+
|           1 | KOSONG  | KOSONG   |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Pada tabel terdapat nama pelanggan dengan nama **[KOSONG]**, jika terjadi pembelian dengan nama pelanggan yang tidak disebutkan maka aplikasi akan memberi nama pelanggan tersebut dengan nama **[KOSONG]**. Nama pelanggan dengan nama **[KOSONG]** ini disebut dengan dummy data.

## KONSEP TRIGGER

Trigger adalah perintah INSERT, UPDATE, DELETE, FUNCTION, PROCEDURE yang ditanam pada MySQL yang akan **DIJALANKAN** pada kejadian berikut:

- AFTER INSERT (setelah INSERT) pada tabel yang dimaksud
- BEFORE INSERT (sebelum INSERT) pada tabel yang dimaksud
- AFTER DELETE (setelah DELETE) pada tabel yang dimaksud
- BEFORE DELETE (Sebelum DELETE) pada tabel yang dimaksud
- AFTER UPDATE (Setelah UPDATE) pada tabel yang dimaksud
- BEFORE UPDATE (Sebelum UPDATE) pada tabel yang dimaksud

Sebelum membuat trigger pastikan anda memahami proses bisnis dari pembuatan database yang anda lakukan. Sebagai latihan buka semua database yang sudah dibuat pada materi sebelumnya, ikuti langkah berikut;

### 1. TAMPILKAN SEMUA TABEL

```
MariaDB [dbtoko]> SHOW TABLES;
+-----+
| Tables_in_dbtoko |
+-----+
| tblbarang        |
| tblkelompok      |
| tblrorder        |
| tblrorderdetail |
| tblpelanggan     |
| view_barang      |
+-----+
6 rows in set (0.01 sec)
```

## 2. LIHAT STRUKTUR SEMUA TABEL

[tblbarang]

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO | PRI | NULL | auto_increment |
| idkelompok | int(11) | YES | MUL | NULL |
| barang | varchar(200) | YES | | NULL |
| stokbarang | int(11) | YES | | NULL |
| hargabeli | float | YES | | NULL |
| hargajual | float | YES | | NULL |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

[tblkelompok]

```
MariaDB [dbtoko]> DESCRIBE tblkelompok;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idkelompok | int(11) | NO | PRI | NULL | auto_increment |
| kelompok | varchar(100) | YES | | NULL |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

[tblorder]

```
MariaDB [dbtoko]> DESCRIBE tblorder;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idorder | int(11) | NO | PRI | NULL | auto_increment |
| idpelanggan | int(11) | YES | MUL | NULL |
| faktur | varchar(50) | YES | | NULL |
| tanggalorder | date | YES | | NULL |
| total | float | YES | | NULL |
| bayar | float | YES | | NULL |
| kembali | float | YES | | NULL |
+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

**[tblorderdetail]**

```
MariaDB [dbtoko]> DESCRIBE tblorderdetail;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idorderdetail | int(11) | NO | PRI | NULL | auto_increment |
| idorder | int(11) | YES | MUL | NULL |
| idbarang | int(11) | YES | MUL | NULL |
| jumlah | int(11) | YES | | NULL |
| hargajual | float | YES | | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

**[tblpelanggan]**

```
MariaDB [dbtoko]> DESCRIBE tblpelanggan;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idpelanggan | int(11) | NO | PRI | NULL | auto_increment |
| nama | varchar(200) | YES | | NULL |
| alamat | varchar(255) | YES | | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

### 3. PENYUSUNAN PROSES BISNIS

kolom [stokbarang] pada tabel [tblbarang] akan **BERKURANG** jika terjadi **INSERT** data pada tabel [tblorderdetail]

kolom [stokbarang] pada tabel [tblbarang] akan **BERTAMBAH** jika terjadi **DELETE** data pada tabel [tblorderdetail]

kolom [total] pada tabel [tblorder] akan **BERTAMBAH** jika terjadi **INSERT** data pada tabel [tblorderdetail]

kolom [total] pada tabel [tblorder] akan **BERKURANG** jika terjadi **DELETE** data pada tabel [tblorderdetail]

### 4. PEMERIKSAAN NAMA KOLOM YANG SAMA

Sebelum membuat TRIGGER pastikan tidak ada nama kolom yang sama pada setiap tabel. Jika melihat tabel diatas terdapat nama kolom yang sama. Yaitu [**hargajual**] pada tabel [tblbarang] dan [**hargajual**] pada tabel [tblorderdetail]. Ubah nama kolom [hargajual] di tabel [tblorderdetail] menjadi [**hargapenjualan**]

```
MariaDB [dbtoko]> ALTER TABLE tblorderdetail CHANGE hargajual hargapenjualan FLOAT;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Lihat hasil perubahan nama kolom

MariaDB [dbtoko]> DESCRIBE tblorderdetail;						
Field	Type	Null	Key	Default	Extra	
idorderdetail	int(11)	NO	PRI	NULL	auto_increment	
idorder	int(11)	YES	MUL	NULL		
idbarang	int(11)	YES	MUL	NULL		
jumlah	int(11)	YES		NULL		
hargapenjualan	float	YES		NULL		

5 rows in set (0.01 sec)

## 5. PEMERIKSAAN KOLOM UNTUK PERUBAHAN HASIL TRIGGER

Lakukan pemeriksaan pada pada kolom yang akan terkena dampak dari proses trigger. Jika yang diproses oleh trigger adalah operasi matematika (penjumlahan, pengurangan, pembagian, perkalian). Pastikan SET DEFAULT VALUE pada kolom tabel tersebut di isi dengan **ANGKA NOL**. Jika belum di isi angka NOL lakukan ALTER untuk merubah kolom tersebut. Lihat contoh berikut;

Kolom [stokbarang] belum di SET DEFAULT dengan ANGKA NOL

MariaDB [dbtoko]> DESCRIBE tblbarang;						
Field	Type	Null	Key	Default	Extra	
idbarang	int(11)	NO	PRI	NULL	auto_increment	
idkelompok	int(11)	YES	MUL	NULL		
barang	varchar(200)	YES		NULL		
stokbarang	int(11)	YES		NULL		
hargabeli	float	YES		NULL		
hargajual	float	YES		NULL		

6 rows in set (0.01 sec)

Perintah ALTER pada tabel [tblbarang]

```
MariaDB [dbtoko]> ALTER TABLE tblbarang ALTER COLUMN stokbarang SET DEFAULT 0;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Hasil perubahan

```
MariaDB [dbtoko]> DESCRIBE tblbarang;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idbarang | int(11) | NO   | PRI   | NULL    | auto_increment |
| idkelompok | int(11) | YES  | MUL   | NULL    |                |
| barang    | varchar(200) | YES  |        | NULL    |                |
| stokbarang | int(11) | YES  |        | 0       |                |
| hargabeli | float   | YES  |        | NULL    |                |
| hargajual  | float   | YES  |        | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Pemeriksaan kolom [**total**] pada tabel [**tblorder**]. Belum di set DEFAULT ANGKA NOL

```
MariaDB [dbtoko]> DESCRIBE tblorder;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idorder | int(11) | NO   | PRI   | NULL    | auto_increment |
| idpelanggan | int(11) | YES  | MUL   | NULL    |                |
| faktur    | varchar(50) | YES  |        | NULL    |                |
| tanggalorder | date   | YES  |        | NULL    |                |
| total     | float   | YES  |        | NULL    |                |
| bayar      | float   | YES  |        | NULL    |                |
| kembali   | float   | YES  |        | NULL    |                |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

Lakukan perubahan dengan ALTER. Perintah ALTER pada tabel [**tblorder**]

```
MariaDB [dbtoko]> ALTER TABLE tblorder ALTER COLUMN total SET DEFAULT 0;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Hasil perubahan kolom [**total**] pada [**tblorder**]

```
MariaDB [dbtoko]> DESCRIBE tblorder;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idorder | int(11) | NO   | PRI   | NULL    | auto_increment |
| idpelanggan | int(11) | YES  | MUL   | NULL    |                |
| faktur    | varchar(50) | YES  |        | NULL    |                |
| tanggalorder | date   | YES  |        | NULL    |                |
| total     | float   | YES  |        | 0       |                |
| bayar      | float   | YES  |        | NULL    |                |
| kembali   | float   | YES  |        | NULL    |                |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

## PEMBUATAN TRIGGER

Setelah semua konsep, proses bisnis, dan pemeriksaan tabel dilakukan sekarang waktunya membuat TRIGGER. Ada 4 trigger yang akan dibuat.

Nama Trigger	Kejadian	Fungsi
kurang_stok	BEFORE INSERT ON	Mengurangi [stokbarang] pada tabel [tblbarang] SEBELUM INSERT pada tabel [tblorderdetail]
tambah_total	AFTER INSERT ON	Menambah [total] pada tabel [tblorder] SESUDAH INSERT pada tabel [tblorderdetail]
tambah_stok	BEFORE DELETE ON	Menambah [stokbarang] pada tabel [tblbarang] SEBELUM DELETE pada tabel [tblorderdetail]
kurang_total	AFTER DELETE ON	Mengurangi [total] pada tabel [tblorder] SESUDAH DELETE pada tabel [tblorderdetail]

```
CREATE TRIGGER nama_trigger
AFTER INSERT ON nama_tabel_yang_dipasang_trigger
FOR EACH ROW
ketik_perintah_trigger_disini_akhiri_dengan_titik_koma;
```

**kurang\_stok**

```
MariaDB [dbtoko]> CREATE TRIGGER kurang_stok
-> BEFORE INSERT ON tblorderdetail
-> FOR EACH ROW
-> UPDATE tblbarang SET stokbarang=stokbarang - NEW.jumlah
-> WHERE idbarang = NEW.idbarang;
Query OK, 0 rows affected (0.04 sec)
```

**PENJELASAN:**

Trigger [kurang\_stok] digunakan untuk mengurangi [stokbarang] pada [tblbarang] saat **BEFORE INSERT** (sebelum insert) pada tabel [tblorderdetail].

**NEW.jumlah** adalah data atau nilai yang **AKAN MASUK** pada kolom [jumlah] di tabel [tblorderdetail]

**Contoh :** jika stok barang 100 kemudian terjadi penjualan pada tabel [tblorderdetail] sebanyak 35 maka nilainya menjadi [stokbarang = 100 – 35] nilai selanjutnya [stokbarang] menjadi 65

### tambah\_total

```
MariaDB [dbtoko]> CREATE TRIGGER tambah_total
    -> AFTER INSERT ON tblorderdetail
    -> FOR EACH ROW
    -> UPDATE tblorder SET
        -> total = total + (NEW.hargapenjualan * NEW.jumlah)
        -> WHERE idorder = NEW.idorder;
Query OK, 0 rows affected (0.02 sec)
```

#### PENJELASAN:

Trigger [**tambah\_total**] digunakan untuk menambah nilai [**total**] pada tabel [**tblorder**] saat **AFTER INSERT** (sesudah insert) pada tabel [**tblorderdetail**].

[**NEW.hargapenjualan \* NEW.jumlah**] adalah data jumlah yang akan di jual dan harga penjualan yang **AKAN MASUK** pada kolom [**jumlah**] dan kolom [**hargapenjualan**] di tabel [**tblorderdetail**]

**Contoh :** terjadi penjualan pada barang dengan [**iDCLrang=1**] sebanyak [**jumlah = 35**] dengan [**hargapenjualan = 12000**] maka [**total=420000**]

### tambah\_stok

```
MariaDB [dbtoko]> CREATE TRIGGER tambah_stok
    -> BEFORE DELETE ON tblorderdetail
    -> FOR EACH ROW
    -> UPDATE tblbarang SET stokbarang=stokbarang + OLD.jumlah
        -> WHERE idbarang = OLD.idbarang;
Query OK, 0 rows affected (0.02 sec)
```

#### PENJELASAN

Trigger [**tambah\_stok**] digunakan untuk menambah [**stokbarang**] pada [**tblbarang**] pada saat **BEFORE DELETE** (sebelum delete) pada tabel [**tblorderdetail**].

**OLD.jumlah** adalah data yang **SUDAH ADA** pada kolom [**jumlah**] di tabel [**tblorderdetail**]

**Contoh :** jika stok barang 65 kemudian terjadi **DELETE** pada tabel [**tblorderdetail**] maka nilai yang [**jumlah = 35**] yang ada pada tabel [**tblorderdetail**] akan dijumlahkan dengan nilai [**stokbarang**] sehingga nilainya menjadi [**stokbarang = 65 + 35**] nilai selanjutnya [**stokbarang**] menjadi 100

### kurang\_total

```
MariaDB [dbtoko]> CREATE TRIGGER kurang_total
    -> AFTER DELETE ON tblorderdetail
    -> FOR EACH ROW
    -> UPDATE tblorder SET
        -> total = total - (OLD.hargapenjualan * OLD.jumlah)
        -> WHERE idorder = OLD.idorder;
Query OK, 0 rows affected (0.02 sec)
```

#### PENJELASAN:

Trigger [**kurang\_total**] digunakan untuk mengurangi nilai [**total**] pada tabel [**tblorder**] saat **AFTER DELETE** (sesudah **DELETE**) pada tabel [**tblorderdetail**].

[**OLD.hargapenjualan \* OLD.jumlah**] adalah data jumlah yang dan harga penjualan yang sudah ada pada tabel [**tblorderdetail**]

## MENAMPILKAN TRIGGER

Trigger yang sudah dibuat bisa ditempilkan dengan cara

```
MariaDB [dbtoko]> SHOW TRIGGERS;
```

Tampilan hasil trigger yang dibuat;

```
| kurang_stok | INSERT | tblorderdetail | UPDATE tblbarang SET
| WHERE idbarang = NEW.idbarang | BEFORE | NULL    | NO_AUTO_CREATE
|           | cp850_general_ci | latin1_swedish_ci |
| tambah_total | INSERT | tblorderdetail | UPDATE tblorder SET t
| WHERE idorder = NEW.idorder | AFTER  | NULL    | NO_AUTO_CREATE
|           | cp850_general_ci | latin1_swedish_ci |
| tambah_stok  | DELETE | tblorderdetail | UPDATE tblbarang SET
| WHERE idbarang = OLD.idbarang | BEFORE | NULL    | NO_AUTO_CREATE
|           | cp850_general_ci | latin1_swedish_ci |
| kurang_total | DELETE | tblorderdetail | UPDATE tblorder SET t
| WHERE idorder = OLD.idorder | AFTER  | NULL    | NO_AUTO_CREATE
|           | cp850_general_ci | latin1_swedish_ci |
```

## PENGUJIAN TRIGGER

Setelah trigger dibuat maka langkah selanjutnya adalah menguji trigger;

### 1. PERSIAPAN

Tampilkan tabel [tblbarang]

```
MariaDB [dbtoko]> SELECT * FROM tblbarang;
+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang      | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+
|      1   |      6   | Beras Rojo Lele |     100   | 10000    | 12000    |
|      2   |      6   | Beras Cianjur  |      50   | 11000    | 14000    |
|      3   |      6   | Beras Medium   |      70   | 8000     | 10000    |
|      4   |      6   | Beras Mahal    |      30   | 23000    | 30000    |
|      5   |      7   | Gula Putih     |      20   | 12000    | 14000    |
|      6   |      7   | Gula Merah     |      10   | 3000     | 5000     |
|      7   |      7   | Gula Batu      |      40   | 2000     | 3000     |
|      8   |      7   | Gula Aren       |      60   | 7000     | 9000     |
|     10  |      9   | Tepung Terigu   |      50   | 4000     | 6000     |
|     11  |      9   | Tepung Tapioka |      15   | 2000     | 3500     |
|     12  |      9   | Tepung Kanji    |      25   | 3500     | 5000     |
+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

Tampilkan tabel [tblpelanggan]

```
MariaDB [dbtoko]> SELECT * FROM tblpelanggan;
+-----+-----+-----+
| idpelanggan | nama      | alamat    |
+-----+-----+-----+
|           1 | KOSONG    | KOSONG    |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Karena data pelanggan hanya satu, tambahkan 2 pelanggan lagi untuk pengujian

```
MariaDB [dbtoko]> INSERT INTO tblpelanggan VALUES ('','komputerkit','sidoarjo');
Query OK, 1 row affected, 1 warning (0.01 sec)
```

```
MariaDB [dbtoko]> INSERT INTO tblpelanggan VALUES ('','Isa','Lamongan');
Query OK, 1 row affected, 1 warning (0.01 sec)
```

Tampilkan kembali data pelanggan

```
MariaDB [dbtoko]> SELECT * FROM tblpelanggan;
+-----+-----+-----+
| idpelanggan | nama      | alamat    |
+-----+-----+-----+
|           1 | KOSONG    | KOSONG    |
|           2 | komputerkit | sidoarjo  |
|           3 | Isa        | Lamongan |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

Tampilkan struktur tabel [tblorder]

```
MariaDB [dbtoko]> DESCRIBE tblorder;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| idorder    | int(11)   | NO   | PRI | NULL    | auto_increment |
| idpelanggan | int(11)   | YES  | MUL | NULL    |                |
| faktur      | varchar(50) | YES  |      | NULL    |                |
| tanggalorder | date      | YES  |      | NULL    |                |
| total       | float     | YES  |      | 0       |                |
| bayar       | float     | YES  |      | NULL    |                |
| kembali    | float     | YES  |      | NULL    |                |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Tampilkan struktur tabel [tblorderdetail]

```
MariaDB [dbtoko]> DESCRIBE tblorderdetail;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idorderdetail | int(11) | NO   | PRI  | NULL    | auto_increment |
| idorder      | int(11) | YES  | MUL  | NULL    |               |
| idbarang     | int(11) | YES  | MUL  | NULL    |               |
| jumlah        | int(11) | YES  |       | NULL    |               |
| hargapenjualan | float   | YES  |       | NULL    |               |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

## 2. PENGUJIAN INSERT PADA TABEL [tblorderdetail]

Buat order atas nama pelanggan [smkrevit] dengan [idpelanggan = 2] pada tabel [tblorder] dengan cara memasukan data sebagai berikut;

```
MariaDB [dbtoko]> INSERT INTO tblorder
-> (idpelanggan, faktur, tanggalorder)
-> VALUES (2, '001', NOW() );
Query OK, 1 row affected, 1 warning (0.01 sec)
```

Periksa dengan perintah sebagai berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblorder;
+-----+-----+-----+-----+-----+-----+-----+
| idorder | idpelanggan | faktur | tanggalorder | total | bayar | kembali |
+-----+-----+-----+-----+-----+-----+-----+
|      1 |          2 | 001  | 2018-03-16 |     0 | NULL |      NULL |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Buat INSERT data pada tabel [tblorderdetail] dengan menggunakan [idorder = 1] sesuai yang ada di tabel [tblorder] dengan [iDCLrang = 1] pada tabel [tblbarang].

Tabel SEBELUM PROSES INSERT pada tabel [tblorderdetail]

```
MariaDB [dbtoko]> SELECT * FROM tblbarang WHERE idbarang=1;
+-----+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang           | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+-----+
|      1 |          6 | Beras Rojo Lele |        100 |    10000 |     12000 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

INSERT DATA pada tabel [tblorderdetail]

```
MariaDB [dbtoko]> INSERT INTO tblorderdetail  
-> (idorder, idbarang, jumlah, hargapenjualan)  
-> VALUES (1, 1, 35, 12000);  
Query OK, 1 row affected (0.01 sec)
```

Hasil pada tabel [tblorderdetail]

```
MariaDB [dbtoko]> SELECT * FROM tblorderdetail;  
+-----+-----+-----+-----+-----+  
| idorderdetail | idorder | idbarang | jumlah | hargapenjualan |  
+-----+-----+-----+-----+-----+  
| 1 | 1 | 1 | 35 | 12000 |  
+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

Hasil pada tabel [tblorder]

```
MariaDB [dbtoko]> SELECT * FROM tblorder;  
+-----+-----+-----+-----+-----+-----+  
| idorder | idpelanggan | faktur | tanggalorder | total | bayar | kembali |  
+-----+-----+-----+-----+-----+-----+  
| 1 | 2 | 001 | 2018-03-16 | 420000 | NULL | NULL |  
+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

Hasil pada tabel [tblbarang]

```
MariaDB [dbtoko]> SELECT * FROM tblbarang WHERE idbarang=1;  
+-----+-----+-----+-----+-----+-----+  
| idbarang | idkelompok | barang | stokbarang | hargabeli | hargajual |  
+-----+-----+-----+-----+-----+-----+  
| 1 | 6 | Beras Rojo Lele | 65 | 10000 | 12000 |  
+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

### 3. PENGUJIAN DELETE PADA TABEL [tblorderdetail]

Lakukan DELETE pada tabel [tblorderdetail] dengan perintah sebagai berikut;

```
MariaDB [dbtoko]> DELETE FROM tblorderdetail  
-> WHERE idorderdetail = 1;  
Query OK, 1 row affected (0.00 sec)
```

Periksa hasil pada tabel [tblorder], setelah dihapus data pada tabel [tblorderdetail]  
kolom [total] akan kembali ke nilai awal

```
MariaDB [dbtoko]> SELECT * FROM tblorder;  
+-----+-----+-----+-----+-----+-----+  
| idorder | idpelanggan | faktur | tanggalorder | total | bayar | kembali |  
+-----+-----+-----+-----+-----+-----+  
| 1 | 2 | 001 | 2018-03-16 | 0 | NULL | NULL |  
+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

Periksa pada tabel [tblbarang], setelah dihapus data pada tabel [tblorderdetail] kolom [stokbarang] akan kembali ke nilai awal

```
MariaDB [dbtoko]> SELECT * FROM tblbarang WHERE idbarang=1;
+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang      | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+
|       1   |        6 | Beras Rojo Lele |      100 |    10000 |     12000 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

## HAPUS TRIGGER

Jika trigger ada yang salah atau tidak digunakan lagi bisa dihapus dengan cara sebagai berikut;

```
MariaDB [dbtoko]> DROP TRIGGER kurang_total;
Query OK, 0 rows affected (0.01 sec)
```

Periksa trigger setelah dihapus;

```
MariaDB [dbtoko]> SHOW TRIGGERS;
```

Hasil pemeriksaan

```
| kurang_stok | INSERT | tblorderdetail | UPDATE tblbarang SET
WHERE idbarang = NEW.idbarang          | BEFORE | NULL    | N
st | cp850           | cp850_general_ci | latin1_swedis
| tambah_total | INSERT | tblorderdetail | UPDATE tblorder SET
total = total + (NEW.hargapenjualan * NEW.jumlah)
WHERE idorder = NEW.idorder | AFTER  | NULL    | NO_AUTO_CREATE_
                           | cp850_general_ci | latin1_swedish_ci |
| tambah_stok  | DELETE | tblorderdetail | UPDATE tblbarang SET
WHERE idbarang = OLD.idbarang          | BEFORE | NULL    | N
st | cp850           | cp850_general_ci | latin1_swedis
```

## JOIN (GABUNGAN TABEL)

Sebelum belajar JOIN buat kembali Trigger yang telah dihapus pada materi diatas

```
MariaDB [dbtoko]> CREATE TRIGGER kurang_total
    -> AFTER DELETE ON tblorderdetail
    -> FOR EACH ROW
    -> UPDATE tblorder SET
    -> total = total - (OLD.hargapenjualan * OLD.jumlah)
    -> WHERE idorder = OLD.idorder;
Query OK, 0 rows affected (0.02 sec)
```

Setelah TRIGGER dibuat kembali lakukan insert data kembali dengan data sebagai berikut;

```
MariaDB [dbtoko]> INSERT INTO tblorderdetail
    -> (idorder, idbarang, jumlah, hargapenjualan)
    -> VALUES (1, 6, 5, 5000);
Query OK, 1 row affected (0.01 sec)
```

```
MariaDB [dbtoko]> INSERT INTO tblorderdetail
    -> (idorder, idbarang, jumlah, hargapenjualan)
    -> VALUES (1, 12, 5, 5000);
Query OK, 1 row affected (0.01 sec)
```

PERIKSA tabel [**tblorderdetail**]

```
MariaDB [dbtoko]> SELECT * FROM tblorderdetail;
+-----+-----+-----+-----+-----+
| idorderdetail | idorder | idbarang | jumlah | hargapenjualan |
+-----+-----+-----+-----+-----+
|          1 |      1 |        1 |      35 |       12000 |
|          2 |      1 |        6 |       5 |        5000 |
|          3 |      1 |       12 |       5 |        5000 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Buat INSERT data pada tabel [**tblorder**] lagi dengan data sebagai berikut;

```
MariaDB [dbtoko]> INSERT INTO tblorder
    -> (idpelanggan, faktur, tanggalorder)
    -> VALUES (1, '002', NOW() );
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Periksa tabel [**tblorder**]

```
MariaDB [dbtoko]> SELECT * FROM tblorder;
+-----+-----+-----+-----+-----+-----+-----+
| idorder | idpelanggan | faktur | tanggalorder | total | bayar | kembali |
+-----+-----+-----+-----+-----+-----+-----+
|      1 |          2 | 001 | 2018-03-16 | 470000 |   0 |     NULL |
|      2 |          1 | 002 | 2018-03-16 |     0 |   0 |     NULL |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Lakukan insert data pada [tblorderdetail]

```
MariaDB [dbtoko]> INSERT INTO tblorderdetail  
-> (idorder, idbarang, jumlah, hargapenjualan)  
-> VALUES (2, 4, 5, 30000 );  
Query OK, 1 row affected (0.01 sec)
```

```
MariaDB [dbtoko]> INSERT INTO tblorderdetail  
-> (idorder, idbarang, jumlah, hargapenjualan)  
-> VALUES (2, 10, 5, 6000 );  
Query OK, 1 row affected (0.03 sec)
```

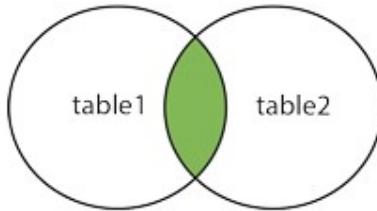
PERIKSA tabel [tblorderdetail]

```
MariaDB [dbtoko]> SELECT * FROM tblorderdetail;  
+-----+-----+-----+-----+-----+  
| idorderdetail | idorder | idbarang | jumlah | hargapenjualan |  
+-----+-----+-----+-----+-----+  
| 1 | 1 | 1 | 35 | 12000 |  
| 2 | 1 | 6 | 5 | 5000 |  
| 3 | 1 | 12 | 5 | 5000 |  
| 4 | 2 | 4 | 5 | 30000 |  
| 5 | 2 | 10 | 5 | 6000 |  
+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

Dari tabel diatas yang masuk ke kolom adalah kode yang sulit dipahami. Untuk memudahkan pembacaan pada tabel maka bisa dibuat gabungan tabel (**JOIN**) untuk memudahkan pembacaan.

## INNER JOIN (MENGAMBIL BAGIAN YANG ADA DI TABEL MASTER DAN TABEL TRANSAKSI)

INNER JOIN



TABEL A = **tblbarang**

TABEL B = **tblorderdetail**

Dari gambar diatas bisa dibuat  
INNER JOIN untuk mengetahui **barang apa yang sudah laku**.

```
SELECT tabel_master.kolom_master, tabel_transaksi.kolom_transaksi
FROM tabel_master
INNER JOIN tabel_transaksi ON tabel_master.kolom_master = tabel_transaksi.kolom_transaksi;
```

JOIN HARUS DIMULAI DARI TABEL MASTER

```
MariaDB [dbtoko]> SELECT tblbarang.barang, tblorderdetail.jumlah,
-> tblorderdetail.hargapenjualan FROM tblbarang
-> INNER JOIN tblorderdetail
-> ON tblbarang.idbarang = tblorderdetail.idbarang;
```

Hasil dari perintah diatas adalah;

barang	jumlah	hargapenjualan
Beras Rojo Lele	35	12000
Gula Merah	5	5000
Tepung Kanji	5	5000
Beras Mahal	5	30000
Tepung Terigu	5	6000

5 rows in set (0.00 sec)

INNER JOIN untuk mengetahui **pelanggan yang melakukan ORDER**

```
MariaDB [dbtoko]> SELECT tblpelanggan.nama,
-> tblorder.tanggalorder, tblorder.total
-> FROM tblpelanggan
-> INNER JOIN tblorder
-> ON tblpelanggan.idpelanggan = tblorder.idpelanggan;
```

Nama pelanggan yang melakukan order

nama	tanggalorder	total
KOSONG	2018-03-16	180000
komputerkit	2018-03-16	470000

2 rows in set (0.00 sec)

### INNER JOIN BANYAK TABEL

Dengan menggunakan INNER JOIN banyak tabel bisa diketahui barang yang dibeli pelanggan dan total pembelian oleh pelanggan.

```
MariaDB [dbtoko]> SELECT tblorder.tanggalorder, tblpelanggan.nama,
-> tblbarang.barang,tblorderdetail.jumlah,
-> tblorderdetail.hargapenjualan, tblorder.total
-> FROM tblpelanggan
-> INNER JOIN tblorder
-> ON tblpelanggan.idpelanggan = tblorder.idpelanggan
-> INNER JOIN tblorderdetail
-> ON tblorder.idorder = tblorderdetail.idorder
-> INNER JOIN tblbarang
-> ON tblorderdetail.idbarang = tblbarang.idbarang
-> ORDER BY tblpelanggan.nama;
```

Hasil INNER JOIN dengan banyak tabel

tanggalorder	nama	barang	jumlah	hargapenjualan	total
2018-03-16	komputerkit	Beras Rojo Lele	35	12000	470000
2018-03-16	komputerkit	Tepung Kanji	5	5000	470000
2018-03-16	komputerkit	Gula Merah	5	5000	470000
2018-03-16	KOSONG	Tepung Terigu	5	6000	180000
2018-03-16	KOSONG	Beras Mahal	5	30000	180000

5 rows in set (0.00 sec)

## LEFT JOIN (MENAMPILKAN TABEL MASTER)

Left join digunakan untuk menampilkan semua yang ada di tabel master dan mengambil sebagian yang ada di tabel transaksi

```
MariaDB [dbtoko]> SELECT tblbarang.barang,
-> tblorderdetail.jumlah, tblorderdetail.hargapenjualan
-> FROM tblbarang
-> LEFT JOIN tblorderdetail
-> ON tblbarang.idbarang = tblorderdetail.idbarang
-> ORDER BY tblorderdetail.jumlah DESC;
```

Hasil LEFT JOIN

barang	jumlah	hargapenjualan
Beras Rojo Lele	35	12000
Tepung Terigu	5	6000
Beras Mahal	5	30000
Tepung Kanji	5	5000
Gula Merah	5	5000
Beras Cianjur	NULL	NULL
Tepung Tapioka	NULL	NULL
Gula Aren	NULL	NULL
Gula Batu	NULL	NULL
Gula Putih	NULL	NULL
Beras Medium	NULL	NULL

11 rows in set (0.00 sec)

## RIGHT JOIN (MENAMPILKAN TABEL TRANSAKSI)

Right Join digunakan untuk menampilkan semua yang ada di tabel transaksi dan sebagian yang ada di tabel master.

```
MariaDB [dbtoko]> SELECT tblbarang.barang,
-> tblorderdetail.jumlah, tblorderdetail.hargapenjualan
-> FROM tblbarang
-> RIGHT JOIN tblorderdetail
-> ON tblbarang.idbarang = tblorderdetail.idbarang
-> ORDER BY tblorderdetail.jumlah DESC;
```

Hasil RIGHT JOIN

barang	jumlah	hargapenjualan
Beras Rojo Lele	35	12000
Gula Merah	5	5000
Tepung Kanji	5	5000
Beras Mahal	5	30000
Tepung Terigu	5	6000

5 rows in set (0.00 sec)

## STORE PROCEDURE

Stor Procedure adalah blok program yang diletakan pada MySQL. Blok yang sudah dibuat bisa dipanggil jika diperlukan. PROCEDURE TIDAK MEMILIKI RETURN

### PROCEDURE TANPA PARAMETER

```
CREATE PROCEDURE nama_procedure  
Isi_procedure;
```

```
MariaDB [dbtoko]> CREATE PROCEDURE namapelanggan()  
-> SELECT * FROM tblpelanggan;  
Query OK, 0 rows affected (0.01 sec)
```

### PEMANGGILAN PROCEDURE

```
CALL nama_procedure;
```

```
MariaDB [dbtoko]> CALL namapelanggan();  
+-----+-----+-----+  
| idpelanggan | nama | alamat |  
+-----+-----+-----+  
| 1 | KOSONG | KOSONG |  
| 2 | komputerkit | sidoarjo |  
| 3 | Isa | Lamongan |  
+-----+-----+-----+  
3 rows in set (0.00 sec)
```

### PROCEDURE DENGAN PARAMETER

```
MariaDB [dbtoko]> CREATE PROCEDURE caribarang (barang VARCHAR(100))  
-> SELECT * FROM tblbarang  
-> WHERE tblbarang.barang LIKE barang;  
Query OK, 0 rows affected (0.02 sec)
```

Panggil procedure yang sudah dibuat dengan parameter yang dimasuka ['%t%'] yang menunjukan nama barang yang mengandung huruf [t]

```
MariaDB [dbtoko]> CALL caribarang ('%t%');
+-----+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+-----+
|      5 |        7 | Gula Putih |       20 |    12000 |    14000 |
|      7 |        7 | Gula Batu  |       40 |     2000 |    3000  |
|     10 |        9 | Tepung Terigu |      45 |     4000 |    6000  |
|     11 |        9 | Tepung Tapioka |      15 |     2000 |    3500  |
|     12 |        9 | Tepung Kanji  |      20 |     3500 |    5000  |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Dengan menggunakan PROCEDURE kita bisa membuat semua QUERAY atau SELECT di dalam procedure sehingga memudahkan dalam pembuatan aplikasi.

## MENAMPILKAN SEMUA PROCEDURE

MySQL bisa menampilkan semua isi procedure dengan perintah berikut;

```
MariaDB [dbtoko]> SHOW PROCEDURE STATUS;
```

Procedure yang sudah dibuat bisa dilihat pada

```
+-----+-----+-----+-----+
| Db      | Name      | Type      | Definer   |
| character_set_client | collation_connection | |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| dbtoko | caribarang | PROCEDURE | root@localhost |
|        | cp850          | cp850_general_ci | |
| dbtoko | namapelanggan | PROCEDURE | root@localhost |
|        | cp850          | cp850_general_ci | |
+-----+-----+-----+-----+
```

## MENGHAPUS STORE PROCEDURE

Untuk menghapus store procedure bisa menggunakan perintah;

```
MariaDB [dbtoko]> DROP PROCEDURE caribarang;
Query OK, 0 rows affected (0.01 sec)
```

Periksa kembali procedure dengan perintah

```
MariaDB [dbtoko]> SHOW PROCEDURE STATUS;
+-----+-----+-----+
| Db      | Name          | Type      | Def
ent | character_set_client | collation_cor
+-----+-----+-----+
| dbtoko | namapelanggan | PROCEDURE | ro
| cp850           | cp850_general
+-----+-----+
```

## FUNCTION

Function adalah blok program yang disimpan di MySQL yang bisa menerima INPUT atau PAREMETER dan MEMILIKI RETURN

### MENYIAPKAN ISI FUNCTION

Sebelum membuat Function harus disiapkan dulu code yang akan dibuat sebagai isi function sehingga hasil dari function sesuai dengan yang diharapkan. Pada materi belajar ini kita akan membuat function untuk menampilkan selisih [**hargajual**] dan [**hargabeli**] pada tabel [**tblbarang**]

Tampilkan tabel [**tblbarang**]

```
MariaDB [dbtoko]> SELECT * FROM tblbarang;
+-----+-----+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang        | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+-----+
| 1         | 6           | Beras Rojo Lele | 65          | 10000    | 12000    |
| 2         | 6           | Beras Cianjur   | 50          | 11000    | 14000    |
| 3         | 6           | Beras Medium    | 70          | 8000     | 10000    |
| 4         | 6           | Beras Mahal     | 25          | 23000    | 30000    |
| 5         | 7           | Gula Putih      | 20          | 12000    | 14000    |
| 6         | 7           | Gula Merah      | 5           | 3000     | 5000     |
| 7         | 7           | Gula Batu       | 40          | 2000     | 3000     |
| 8         | 7           | Gula Aren       | 55          | 7000     | 9000     |
| 10        | 9           | Tepung Terigu   | 45          | 4000     | 6000     |
| 11        | 9           | Tepung Tapioka  | 15          | 2000     | 3500     |
| 12        | 9           | Tepung Kanji    | 20          | 3500     | 5000     |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.01 sec)
```

Pada tabel diatas akan ditambahkan 1 kolom lagi sebagai selisih dari [**hargajual**] dikurangi dengan [**hargabeli**]

```
MariaDB [dbtoko]> SELECT *, hargajual - hargabeli AS selisih FROM tblbarang;
+-----+-----+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang | stokbarang | hargabeli | hargajual | selisih |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 6 | Beras Rojo Lele | 65 | 10000 | 12000 | 2000 |
| 2 | 6 | Beras Cianjur | 50 | 11000 | 14000 | 3000 |
| 3 | 6 | Beras Medium | 70 | 8000 | 10000 | 2000 |
| 4 | 6 | Beras Mahal | 25 | 23000 | 30000 | 7000 |
| 5 | 7 | Gula Putih | 20 | 12000 | 14000 | 2000 |
| 6 | 7 | Gula Merah | 5 | 3000 | 5000 | 2000 |
| 7 | 7 | Gula Batu | 40 | 2000 | 3000 | 1000 |
| 8 | 7 | Gula Aren | 55 | 7000 | 9000 | 2000 |
| 10 | 9 | Tepung Terigu | 45 | 4000 | 6000 | 2000 |
| 11 | 9 | Tepung Tapioka | 15 | 2000 | 3500 | 1500 |
| 12 | 9 | Tepung Kanji | 20 | 3500 | 5000 | 1500 |
+-----+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

Selisih [**hargajual**] dan [**hargabeli**] ini disebut dengan [**laba**]. Sekarang akan diambil selisih perbarang untuk dihitung sebagai laba. Sebagai contoh diambil satu barang dengan [**IDCrlang = 1**] maka selisihnya akan ditampilkan seperti gambar dibawah.

```
MariaDB [dbtoko]> SELECT hargajual - hargabeli AS laba
    -> FROM tblbarang WHERE idbarang = 1;
+-----+
| laba |
+-----+
| 2000 |
+-----+
1 row in set (0.00 sec)
```

Setelah isi dibuat, saatnya membuat function. Cara membuat function sebagai berikut;

<b>CREATE FUNCTION nama_function (parameter TIPE DATA) RETURNS TIPE DATA RETURN (isi_function);</b>
---

Isi parameter tipe data sesuai dengan data yang dimasukan. Pada contoh diatas parameter yang dimasukan adalah [**IDCrlang**] dengan tipe data INT. Output yang dihasilkan dari isi function diatas adalah selisih [**hargajual**] – [**hargabeli**] yang bertipe data FLOAT;

```
MariaDB [dbtoko]> CREATE FUNCTION laba (id INT) RETURNS FLOAT
    -> RETURN
    -> (
    -> SELECT hargajual - hargabeli as laba
    -> FROM tblbarang
    -> WHERE idbarang = id
    -> );
Query OK, 0 rows affected (0.02 sec)
```

## MEMANGGIL FUNCTION

Untuk memanggil function gunakan perintah berikut;

```
SELECT FUNCTION nama_function (parameter);
```

```
MariaDB [dbtoko]> SELECT laba(1);
+-----+
| laba(1) |
+-----+
|    2000 |
+-----+
1 row in set (0.00 sec)
```

## PENGGUNAAN FUNCTION

Materi berikut akan menunjukkan penggunaan function pada aplikasi yang sedang dibuat.

Lihat struktur tabel [tblorderdetail]

```
MariaDB [dbtoko]> DESCRIBE tblorderdetail;
+-----+-----+-----+-----+-----+-----+
| Field      | Type     | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| idorderdetail | int(11) | NO   | PRI | NULL    | auto_increment |
| idorder     | int(11) | YES  | MUL | NULL    |                |
| idbarang    | int(11) | YES  | MUL | NULL    |                |
| jumlah      | int(11) | YES  |      | NULL    |                |
| hargapenjualan | float   | YES  |      | NULL    |                |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

Tambahkan sebuah kolom dengan nama [laba]

```
MariaDB [dbtoko]> ALTER TABLE tblorderdetail ADD laba FLOAT;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Periksa kembali dengan describe

```
MariaDB [dbtoko]> DESCRIBE tblorderdetail;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idorderdetail | int(11) | NO   | PRI | NULL    | auto_increment |
| idorder      | int(11) | YES  | MUL | NULL    |                |
| idbarang     | int(11) | YES  | MUL | NULL    |                |
| jumlah       | int(11) | YES  |      | NULL    |                |
| hargapenjualan | float  | YES  |      | NULL    |                |
| laba          | float  | YES  |      | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

Lakukan proses order dengan melakukan insert pada tabel [tblorder]

```
MariaDB [dbtoko]> INSERT INTO tblorder
-> (idpelanggan, faktur, tanggalorder)
-> VALUES (3,'003', NOW());
Query OK, 1 row affected, 1 warning (0.01 sec)
```

Periksa tabel [tblorder]

```
MariaDB [dbtoko]> SELECT * FROM tblorder;
+-----+-----+-----+-----+-----+-----+
| idorder | idpelanggan | faktur | tanggalorder | total | bayar | kembali |
+-----+-----+-----+-----+-----+-----+
| 1 | 2 | 001 | 2018-03-16 | 470000 | NULL | NULL |
| 2 | 1 | 002 | 2018-03-16 | 180000 | NULL | NULL |
| 3 | 3 | 003 | 2018-03-17 | 0 | NULL | NULL |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Buat insert pada tabel [tblorderdetail]

```
MariaDB [dbtoko]> INSERT INTO tblorderdetail
-> (idorder, idbarang, jumlah, hargapenjualan, laba)
-> VALUES (3, 8, 5, 9000, (SELECT laba(8)));
Query OK, 1 row affected (0.01 sec)
```

Pada saat INSERT kolom [**laba**] akan diisi dengan function yang menghitung [**hargajual**] dikurangi [**hargabeli**] dimana [**iDCLrang**] dimasukan sebagai parameter input pada function.

Periksa tabel [tblorderdetail]

```
MariaDB [dbtoko]> SELECT * FROM tblorderdetail ORDER BY laba DESC;
+-----+-----+-----+-----+-----+-----+
| idorderdetail | idorder | idbarang | jumlah | hargapenjualan | laba |
+-----+-----+-----+-----+-----+-----+
|       6 |      3 |        8 |       5 |         9000 |   2000 |
|       1 |      1 |        1 |      35 |        12000 |    NULL |
|       2 |      1 |        6 |       5 |         5000 |    NULL |
|       3 |      1 |       12 |       5 |         5000 |    NULL |
|       4 |      2 |        4 |       5 |        30000 |    NULL |
|       5 |      2 |       10 |       5 |         6000 |    NULL |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Periksa selisih [hargajual] – [hargabeli] pada tabel [tblbarang]. Jika sesuai berarti function sudah berjalan sesuai dengan yang diharapkan.

```
MariaDB [dbtoko]> SELECT * FROM tblbarang WHERE idbarang = 8;
+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+
|       8 |        7 | Gula Aren |       55 |       7000 |     9000 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

## MENAMPILKAN SEMUA FUNCTION

Untuk menampilkan function yang sudah dibuat bisa menggunakan

```
MariaDB [dbtoko]> SHOW FUNCTION STATUS;
```

Tampilan function yang telah dibuat

```
+-----+-----+
| Db      | Name   | Type   | Definer
acther_set_client | collation_connection | Dat
+-----+-----+
+-----+-----+
| dbtoko | laba   | FUNCTION | root@localhost
0          | cp850_general_ci   | laba
+-----+-----+
```

## MENGHAPUS FUNCTION

Untuk menghapus function bisa menggunakan perintah berikut;

```
MariaDB [dbtoko]> DROP FUNCTION laba;
Query OK, 0 rows affected (0.02 sec)
```

Periksa kembali function yang telah dihapus

```
MariaDB [dbtoko]> SHOW FUNCTION STATUS;
Empty set (0.01 sec)
```

## **SELECT AGGREGATE**

SELECT AGREGATE adalah SELECT yang menampilkan function bawaan dari MySQL. Ada beberapa select aggregate yang akan dipelajari yaitu;

<b>MIN</b>	Menampilkan nilai TERKECIL pada kolom yang dimaksud
<b>MAX</b>	Menampilkan nilai TERBESAR pada kolom yang dimaksud
<b>SUM</b>	Menampilkan nilai PENJUMLAHAN pada kolom yang dimaksud
<b>AVG</b>	Menampilkan nilai RATA - RATA pada kolom yang dimaksud
<b>COUNT</b>	Menampilkan JUMLAH BARIS pada kolom yang dimaksud
<b>COUNT(*)</b>	Menampilkan JUMLAH BARIS pada TABEL

```
SELECT function_aggregate(nama_kolom);
ANTARA NAMA AGGREGATE DAN KURUNG TIDAK BOLEH ADA SPASI
```

SELECT MIN

```
MariaDB [dbtoko]> SELECT MIN(hargajual) FROM tblbarang;
+-----+
| MIN(hargajual) |
+-----+
|      3000 |
+-----+
1 row in set (0.00 sec)
```

SELECT MAX

```
MariaDB [dbtoko]> SELECT MAX(hargajual) FROM tblbarang;
+-----+
| MAX(hargajual) |
+-----+
|     30000 |
+-----+
1 row in set (0.00 sec)
```

..

SELECT SUM

```
MariaDB [dbtoko]> SELECT SUM(hargajual) FROM tblbarang;
+-----+
| SUM(hargajual) |
+-----+
|     111500 |
+-----+
1 row in set (0.00 sec)
```

```
SELECT AVG
```

```
MariaDB [dbtoko]> SELECT AVG(hargajual) FROM tblbarang;
+-----+
| AVG(hargajual) |
+-----+
| 10136.3636363636 |
+-----+
1 row in set (0.00 sec)
```

```
SELECT COUNT
```

```
MariaDB [dbtoko]> SELECT COUNT(hargajual) FROM tblbarang;
+-----+
| COUNT(hargajual) |
+-----+
|           11 |
+-----+
1 row in set (0.00 sec)
```

```
SELECT COUNT(*)
```

```
MariaDB [dbtoko]> SELECT COUNT(*) FROM tblbarang;
+-----+
| COUNT(*) |
+-----+
|       11 |
+-----+
1 row in set (0.00 sec)
```

## SELECT BETWEEN (SELECT ANTARA DUA NILAI)

Select yang digunakan untuk menampilkan data antara dua nilai terendah dan tertinggi;

```
SELECT nama_kolom_atau_* FROM nama_tabel WHERE nama_kolom BETWEEN awal  
AND akhir
```

```
MariaDB [dbtoko]> SELECT * FROM tblbarang  
    -> WHERE hargajual BETWEEN 5000 AND 15000;
+-----+-----+-----+-----+-----+-----+
| idbarang | idkelompok | barang      | stokbarang | hargabeli | hargajual |
+-----+-----+-----+-----+-----+-----+
|     1   |      6   | Beras Rojo Lele |      65   | 10000   | 12000   |
|     2   |      6   | Beras Cianjur   |      50   | 11000   | 14000   |
|     3   |      6   | Beras Medium    |      70   | 8000    | 10000   |
|     5   |      7   | Gula Putih      |      20   | 12000   | 14000   |
|     6   |      7   | Gula Merah      |      5    | 3000    | 5000    |
|     8   |      7   | Gula Aren        |      55   | 7000    | 9000    |
|    10  |      9   | Tepung Terigu    |      45   | 4000    | 6000    |
|    12  |      9   | Tepung Kanji     |      20   | 3500    | 5000    |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

## **SELECT DISTINCT (MENAMPILKAN DATA YANG SAMA HANYA SATU KALI)**

Untuk menampilkan data yang sama satu kali.

```
SELECT DISTINCT nama_kolom FROM nama_tabel_view
```

```
MariaDB [dbtoko]> SELECT DISTINCT kelompok FROM view_barang;
+-----+
| kelompok |
+-----+
| Beras   |
| Gula    |
| Tepung  |
+-----+
3 rows in set (0.00 sec)
```

## **START TRANSACTION, COMMIT, DAN ROLLBACK**

Setiap perintah di MySQL yang masuk kelompok (INSERT, UPDATE, DELETE, SELECT) bisa dimasukan kedalam perintah START TRANSACTION, COMMIT, DAN ROLLBACK. Perintah yang dimulai dengan START TRANSACTION bisa dilakukan ROLLBACK (undo atau pembatalan perintah). COMMIT digunakan agar perintah yang dijalankan **TIDAK BISA DI ROLLBACK (undo)**.

buka kembali tabel [tblkelompok];

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
|      10 | Minyak  |
|      11 | Jajan   |
|      12 | Roti Basah |
+-----+-----+
6 rows in set (0.00 sec)
```

Lakukan perintah berikut, untuk memulai;

```
MariaDB [dbtoko]> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

Berikan perintah INSERT sebagai berikut;

```
MariaDB [dbtoko]> INSERT INTO tblkelompok VALUES ('','Minuman');
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Periksa data yang sudah dimasukan;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
|      10 | Minyak  |
|      11 | Jajan   |
|      12 | Roti Basah |
|      14 | Minuman |
+-----+-----+
7 rows in set (0.00 sec)
```

Berikan perintah berikut;

```
MariaDB [dbtoko]> ROLLBACK;
Query OK, 0 rows affected (0.01 sec)
```

Periksa kembali data yang sudah dimasukan

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
|      10 | Minyak  |
|      11 | Jajan   |
|      12 | Roti Basah |
+-----+-----+
6 rows in set (0.00 sec)
```

Ulangi perintah INSERT sebelumnya;

```
MariaDB [dbtoko]> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

MariaDB [dbtoko]> INSERT INTO tblkelompok VALUES ('','Minuman');
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Dan ikuti dengan perintah;

```
MariaDB [dbtoko]> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

Periksa data yang sudah dimasukan;

```
MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
|      10 | Minyak  |
|      11 | Jajan   |
|      12 | Roti Basah |
|      15 | Minuman |
+-----+-----+
7 rows in set (0.00 sec)
```

Lakukan ROLLBACK dan periksa hasilnya. **TRANSAKSI YANG SUDAH DI COMMIT TIDAK BISA DI ROLLBACK (undo).**

```
MariaDB [dbtoko]> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

MariaDB [dbtoko]> SELECT * FROM tblkelompok;
+-----+-----+
| idkelompok | kelompok |
+-----+-----+
|       6 | Beras   |
|       7 | Gula    |
|       9 | Tepung  |
|      10 | Minyak  |
|      11 | Jajan   |
|      12 | Roti Basah |
|      15 | Minuman |
+-----+-----+
7 rows in set (0.00 sec)
```

## **KESIMPULAN START TRANSACTION, COMMIT, & ROLLBACK**

- Transaksi adalah perintah (INSERT, UPDATE, DELETE, SELECT)
- Transaksi yang bisa di ROLLBACK (Undo) adalah transaksi yang dimulai dengan START TRANSACTION dan BELUM DI AKHIRI DENGAN COMMIT
- Transaksi yang sudah di COMMIT TIDAK BISA di ROLLBACK

## **PENGGUNAAN START TRANSACTION, COMMIT, & ROLLBACK**

Ketika database yang anda buat sudah digunakan oleh orang yang membeli program ,aka database tersebut akan terisi dengan data VALID (BENAR) sesuai dengan yang dimiliki oleh orang tersebut. Kemudian database tersebut dikembalikan untuk proses modifikasi atau penambahan sesuai dengan kebutuhan orang tersebut. Maka data yang VALID dari database TIDAK BOLEH dicampur dengan DATA DUMMY (data coba - coba) karena anda sedang melakukan PENGUJIAN database. Agar data DUMMY tidak tercampur dengan data VALID gunakan perintah START TRANSACTION sebelum melakukan proses TRANSAKSI. jika pengujian sudah selesai lakukan ROLLBACK. Jangan pernah lakukan COMMIT jika data yang dimasukan data DUMMY.

# DCL

*Data Control Language*

Manajemen User dan Hak Akses

## TENTANG DCL (Data Control Language)

Pengaturan USER diperlukan agar database yang sudah dibuat dengan susah payah tidak di acak – acak oleh orang yang tidak berhak.

DCL adalah materi yang akan digunakan untuk mempelajari tentang menajemen atau pengelolaan USER dan Hak Akses yang akan menggunakan database MySQL.

Sebelum menggunakan MySQL anda di haruskan login terlebih dahulu dengan user awal [root] dan password [kosong].

```
C:\xampp\mysql\bin>mysql -u root -p  
Enter password:
```

## MENAMPILKAN SEMUA USER

Untuk menampilkan semua user yang audah ada di MySQL gunakan perintah berikut;

```
MariaDB [(none)]> SELECT user, host, password FROM mysql.user;  
+-----+-----+  
| user | host   | password |  
+-----+-----+  
| root | localhost |  
| root | 127.0.0.1 |  
| root | ::1 |  
|      | localhost |  
| pma  | localhost |  
+-----+-----+  
5 rows in set (0.00 sec)
```

**HOST** adalah lokasi komputer dimana database MySQL disimpan. Jika menggunakan komputer atau laptop sendiri maka HOST nya adalah localhost atau dengan menggunakan alamat IP 12.7.0.0.1

Penambahan user dan hak akses hanya bisa diberikan oleh user [root]
---

## MENAMBAH USER

Untuk menambah user gunakan perintah berikut;

```
CREATE USER 'nama_user'@'host';
```

jika pada komputer di lokasi yang lain misal pada server jaringan atau internet masukan IP dari SERVER tersebut pada **HOST**

```
MariaDB [(none)]> CREATE USER 'komputerkit'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

Periksa dengan perintah;

```
MariaDB [(none)]> SELECT user, host, password FROM mysql.user;
+-----+-----+-----+
| user | host | password |
+-----+-----+-----+
| root | localhost |          |
| root | 127.0.0.1 |          |
| root | ::1 |          |
|       | localhost |          |
| pma | localhost |          |
| komputerkit | localhost |          |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

## MEMBERIKAN PASSWORD PADA USER

User yang dibuat di awal BELUM ada passwordnya. Untuk memberikan password gunakan perintah berikut;

```
SET PASSWORD FOR 'nama_user'@'host' = PASSWORD ('password_user');
```

```
MariaDB [(none)]> SET PASSWORD FOR 'komputerkit'@'localhost' = PASSWORD ('komputerkit');
Query OK, 0 rows affected (0.00 sec)
```

Berikan perintah berikut agar MySQL membaca ulang (REFRESH) user dan password yang telah dibuat atau diubah.

```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

## MENGUJI USER DAN PASSWORD YANG TELAH DIBUAT

Lakukan quit untuk keluar dari MySQL

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

Masuk lagi dengan perintah, user [**smkrevit**] password [**smkrevit**];

```
C:\xampp\mysql\bin>mysql -u komputerkit -p  
Enter password: *****
```

Periksa database

```
MariaDB [(none)]> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| test |  
+-----+  
2 rows in set (0.00 sec)
```

Pada saat menampilkan database tidak terdapat database seperti pada waktu masuk dengan user [**root**]

## MEMBERIKAN HAK AKSES (**PRIVILEGES**) USER KE DATABASE

Sebelum memberikan hak akses pada database pastikan anda **QUIT** terlebih dahulu, dan masuk kembali dengan user [**root**].

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

Masuk kembali dengan user [**root**]

```
C:\xampp\mysql\bin>mysql -u root -p  
Enter password:
```

Untuk memberikan hak akses pada database gunakan perintah;

```
GRANT ALL PRIVILEGES ON nama_database.* TO 'nama_user'@'host' IDENTIFIED BY  
'password_user';
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON dbtoko.* TO 'komputerkit'@'localhost' IDENTIFIED BY 'komputerkit';  
Query OK, 0 rows affected (0.00 sec)
```

Setelah pemberian hak akses lakukan refresh dengan perintah;

```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Untuk memeriksa apakah hak akses berhasil diberikan anda harus quit terelebih dahulu dari user [root]

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

Masuk kembali dengan user [smkrevit] password [smkrevit]

```
C:\xampp\mysql\bin>mysql -u komputerkit -p
Enter password: *****
```

Tampilkan database dengan perintah;

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database      |
+-----+
| dbtoko        |
| information_schema |
| test          |
+-----+
3 rows in set (0.00 sec)
```

## MENAMPILKAN USER YANG SEDANG LOGIN (MASUK)

Untuk melihat user yang sedang login, gunakan perintah berikut;

```
MariaDB [(none)]> SELECT USER(),CURRENT_USER();
+-----+-----+
| USER()           | CURRENT_USER()   |
+-----+-----+
| komputerkit@localhost | komputerkit@localhost |
+-----+-----+
1 row in set (0.00 sec)
```

## MENAMPILKAN HAK AKSES YANG DIBERIKAN

Untuk memeriksa hak akses yang diberikan pada user, lakukan quit terlebih dahulu dan login kembali dengan user [root]

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

Login dengan user [root]

```
C:\xampp\mysql\bin>mysql -u root -p  
Enter password:
```

Gunakan perintah berikut untuk menampilkan hak akses

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

```
MariaDB [(none)]> SHOW GRANTS FOR 'komputerkit'@'localhost';
```

Hasilnya adalah;

```
Grants for komputerkit@localhost  
-----  
GRANT USAGE ON *.* TO 'komputerkit'@'localhost' IDENTIFIED BY PASSWORD '*4B8A2F9E7  
GRANT ALL PRIVILEGES ON `dbtoko`.* TO 'komputerkit'@'localhost'
```

## MENGHAPUS HAK AKSES

Untuk menghapus hak akses pada user [smkrevit] anda harus login sebagai user [root] gunakan perintah sebagai berikut;

Periksa dulu user login anda;

```
MariaDB [(none)]> SELECT USER(), CURRENT_USER();  
+-----+-----+  
| USER() | CURRENT_USER() |  
+-----+-----+  
| root@localhost | root@localhost |  
+-----+-----+  
1 row in set (0.00 sec)
```

Untuk menghapus hak akses gunakan perintah;

```
REVOKE ALL PRIVILEGES ON nama_database.* FROM 'nama_user'@'localhost';
```

```
MariaDB [(none)]> REVOKE ALL PRIVILEGES ON dbtoko.* FROM 'komputerkit'@'localhost';  
Query OK, 0 rows affected (0.00 sec)
```

Refresh perubahan hak akses dengan perintah;

```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Periksa hak akses yang sudah dihapus dengan perintah;

```
MariaDB [(none)]> SHOW GRANTS FOR 'komputerkit'@'localhost';
```

Hasil setelah hak akses dihapus;

```
Grants for komputerkit@localhost
-----
GRANT USAGE ON *.* TO 'komputerkit'@'localhost' IDENTIFIED BY PASSWORD '*4B
```

## MEMBERIKAN HAK AKSES PADA DATABASE DENGAN TABEL

Gunakan perintah berikut untuk memberikan hak akses pada 1 tabel;

```
GRANT SELECT ON nama_database.nama_tabel TO 'nama_user'@'localhost'
IDENTIFIED BY 'password_user';
```

```
MariaDB [(none)]> GRANT ALL ON dbtoko.tblkelompok TO 'komputerkit'@'localhost' IDENTIFIED BY 'komputerkit';
Query OK, 0 rows affected (0.00 sec)
```

Periksa hak akses yang diberikan dengan perintah;

```
MariaDB [(none)]> SHOW GRANTS FOR 'komputerkit'@'localhost';
-----
Grants for komputerkit@localhost
-----
GRANT USAGE ON *.* TO 'komputerkit'@'localhost' IDENTIFIED BY PASSWORD '*4B
GRANT ALL PRIVILEGES ON `dbtoko`.`tblkelompok` TO 'komputerkit'@'localhost'
-----
2 rows in set (0.00 sec)
```

Untuk menambah tabel yang diberikan hak aksesnya gunakan perintah;

```
MariaDB [(none)]> GRANT ALL ON dbtoko.tblbarang TO 'komputerkit'@'localhost' IDENTIFIED BY 'komputerkit';
Query OK, 0 rows affected (0.00 sec)
```

Periksa hasilnya dengan perintah;

```
MariaDB [(none)]> SHOW GRANTS FOR 'komputerkit'@'localhost';
+-----+
| Grants for komputerkit@localhost
+-----+
| GRANT USAGE ON *.* TO 'komputerkit'@'localhost' IDENTIFIED BY PASSWORD '*4B8A
| GRANT ALL PRIVILEGES ON `dbtoko`.`tblkelompok` TO 'komputerkit'@'localhost'
| GRANT ALL PRIVILEGES ON `dbtoko`.`tblbarang` TO 'komputerkit'@'localhost'
+-----+
3 rows in set (0.00 sec)
```

Untuk memeriksa hasil pemberian hak akses, lakukan quit dari user [**root**] dan login dengan user [**smkrevit**]

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

Login dengan user [**smkrevit**]

```
C:\xampp\mysql\bin>mysql -u komputerkit -p
Enter password: *****
```

Tampilkan database

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database
+-----+
| dbtoko
| information_schema
| test
+-----+
3 rows in set (0.01 sec)
```

Aktifkan database [**dbtoko**]

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

Periksa tabel dengan perintah;

```
MariaDB [dbtoko]> SHOW TABLES;
+-----+
| Tables_in_dbtoko
+-----+
| tblbarang
| tblkelompok
+-----+
2 rows in set (0.00 sec)
```

## **MEMBERIKAN HAK AKSES PADA SELECT, INSERT, DELETE, UPDATE, PADA TABEL**

MySQL memberikan pengaturan agar user bisa diberikan hak akses tertentu sehingga dapat menjamin keamanan database.

Untuk memberikan hak akses SELECT, INSERT, DELETE, & UPDATE lakukan perintah berikut;

Lakukan QUIT terlebih dahulu

```
MariaDB [dbtoko]> quit;  
Bye
```

Login dengan user [root]

```
C:\xampp\mysql\bin>mysql -u root -p  
Enter password:
```

Berikan perintah berikut untuk memberikan hak akses SELECT DAN INSERT pada tabel [tblpelanggan].

```
GRANT SELECT,INSERT,UPDATE,DELETE ON nama_database.nama_tabel TO  
'nama_user'@'localhost';
```

Jika setelah GRANT adalah ALL artinya user tersebut diberikan hak untuk SELECT INSERT DELETE UPDATE pada tabel tersebut.

```
MariaDB [(none)]> GRANT SELECT,INSERT ON  
      -> dbtoko.tblpelanggan TO 'komputerkit'@'localhost';  
Query OK, 0 rows affected (0.00 sec)
```

Lakukan pengujian dengan cara sebagai berikut;

Lakukan quit terlebih dahulu.

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

Login dengan user [smkrevit]

```
C:\xampp\mysql\bin>mysql -u komputerkit -p  
Enter password: *****
```

Aktifkan [dbtoko]

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

Periksa tabel

```
MariaDB [dbtoko]> SHOW TABLES;
+-----+
| Tables_in_dbtoko |
+-----+
| tblbarang        |
| tblkelompok      |
| tblpelanggan     |
+-----+
3 rows in set (0.00 sec)
```

Lihat struktur tabel [tblpelanggan]

```
MariaDB [dbtoko]> DESCRIBE tblpelanggan;
+-----+-----+-----+-----+-----+-----+
| Field    | Type     | Null | Key | Default | Extra   |
+-----+-----+-----+-----+-----+-----+
| idpelanggan | int(11) | NO  | PRI | NULL    | auto_increment |
| nama       | varchar(200) | YES |     | NULL    |           |
| alamat     | varchar(255) | YES |     | NULL    |           |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Isi tabel pelanggan dengan data berikut;

```
MariaDB [dbtoko]> INSERT INTO tblpelanggan
-> VALUES('','jONI', 'Sidoarjo');
Query OK, 1 row affected, 1 warning (0.01 sec)
```

Periksa dengan perintah berikut;

```
MariaDB [dbtoko]> SELECT * FROM tblpelanggan;
+-----+-----+-----+
| idpelanggan | nama      | alamat    |
+-----+-----+-----+
|          1 | KOSONG    | KOSONG    |
|          2 | komputerkit | sidoarjo  |
|          3 | Isa        | Lamongan  |
|          4 | jONI       | Sidoarjo  |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

Lakukan perintah DELETE pada tabel [tblpelanggan]

```
MariaDB [dbtoko]> DELETE FROM tblpelanggan
-> WHERE idpelanggan = 4;
ERROR 1142 (42000): DELETE command denied to user 'komputerkit'@'localhost' for table 'tblpelanggan'
```

Data tidak bisa dihapus untuk user [smkrevit] karena tidak diberikan hak aksesnya.

Lakukan perintah UPDATE

```
MariaDB [dbtoko]> UPDATE tblpelanggan  
-> SET nama='TEJO' WHERE idpelanggan=4;  
ERROR 1142 (42000): UPDATE command denied to user 'komputerkit'@'localhost' for table 'tblpelanggan'
```

Data tidak bisa diubah untuk user [**smkrevit**] karena tidak diberikan hak aksesnya.

## UBAH PASSWORD USER

Untuk merubah password harus dilakukan oleh user [**root**]

Lakukan quit jika user yang digunakan bukan [**root**], login sebagai user [**root**]

```
C:\xampp\mysql\bin>mysql -u root -p  
Enter password:
```

Ubah password dengan perintah berikut;

```
MariaDB [(none)]> SET PASSWORD FOR 'komputerkit'@'localhost' = PASSWORD('123456');  
Query OK, 0 rows affected (0.00 sec)
```

Refresh dengan perintah berikut;

```
MariaDB [(none)]> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.00 sec)
```

## HAPUS USER

Untuk menghapus user hanya bisa dilakukan oleh user [**root**].

Hapus user dengan perintah berikut;

```
MariaDB [(none)]> DROP USER 'komputerkit'@'localhost';  
Query OK, 0 rows affected (0.00 sec)
```

Periksa user yang telah dihapus dengan perintah berikut;

```
MariaDB [(none)]> SELECT user, password, host FROM mysql.user;  
+-----+-----+-----+  
| user | password | host   |  
+-----+-----+-----+  
| root |          | localhost |  
| root |          | 127.0.0.1 |  
| root |          | ::1      |  
|      |          | localhost |  
| pma  |          | localhost |  
+-----+-----+-----+  
5 rows in set (0.00 sec)
```

## MERUBAH PASSWORD USER [root]

Untuk merubah password user root gunakan perintah berikut

Pastikan sudah keluar dari MySQL sebelum merubah password user [root]

BERIKAN PERINTAH BERIKUT UNTUK SET PASSWORD, JIKA PASSWORD AWAL [**kosong**]

```
C:\xampp\mysql\bin>mysqladmin -u root password 123456
```

Lakukan login dengan user [**root**] dan password [**123456**]

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password: *****
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 15
Server version: 10.1.30-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

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

Berikan perintah berikut jika ingin mengganti password lama dengan password baru

```
MYSQLADMIN -u root -p_paswword_lama PASSWORD
```

```
C:\xampp\mysql\bin>mysqladmin -u root -p123456 password
New password:
Confirm new password:
```

End Of Book-----SEMOGA BUKUNYA BERMANFAAT -----

## **DAFTAR PUSTAKA**

<https://mariadb.org/>

<https://stackoverflow.com/questions/22774739/change-mysql-user-password-using-command-line>

<https://brainly.co.id/tugas/9808222>

<https://www.howtoforge.com/setting-changing-resetting-mysql-root-passwords>