

## MODUL 8

### TRIGGER

#### A. TUJUAN

- Memahami konsep dasar trigger di dalam basis data.
- Memahami implementasi trigger sebagai bentuk respon atas suatu kejadian.
- Mampu menyelesaikan kasus-kasus manipulasi data yang kompleks dengan memanfaatkan trigger.

#### B. PETUNJUK

- Awali setiap aktivitas dengan do'a, semoga berkah dan mendapat kemudahan.
- Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas praktikum dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas

#### C. DASAR TEORI

##### 1. Trigger

Trigger dapat didefinisikan sebagai himpunan kode (prosedural) yang dieksekusi secara otomatis sebagai respon atas suatu kejadian berkaitan dengan tabel basis data. Kejadian (event) yang dapat membangkitkan trigger umumnya berupa pernyataan INSERT, UPDATE, dan DELETE. Berdasarkan ruang lingkungannya, trigger diklasifikasikan menjadi dua jenis: row trigger dan statement trigger. Trigger baris (row) mendefinisikan aksi untuk setiap baris tabel; trigger pernyataan hanya berlaku untuk setiap pernyataan INSERT, UPDATE, atau DELETE.

Dari sisi perilaku (behavior) eksekusi, trigger dapat dibedakan menjadi beberapa jenis; namun umumnya ada dua jenis: trigger BEFORE dan AFTER. Sesuai penamaannya, jenis-jenis ini merepresentasikan waktu eksekusi trigger—misalnya sebelum ataukah sesudah pernyataan- pernyataan yang berkorespondensi.

Adakalanya trigger dipandang sebagai bentuk spesifik dari stored procedure (terkait pendefinisian body). Bagaimanapun, trigger akan dipanggil (secara otomatis) ketika event terjadi, sedangkan stored procedure harus dipanggil secara eksplisit.

## 2. Trigger MySQL

MySQL mendukung fitur trigger termasuk juga stored procedure dan view sejak versi 5.0.2. Sebagaimana objek-objek lainnya, trigger diciptakan menggunakan pernyataan CREATE.

Sintaks pendefinisian trigger diperlihatkan sebagai berikut:

```
CREATE
[DEFINER = { user | CURRENT_USER }]
TRIGGER trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW trigger_stmt
```

MySQL tidak mengizinkan multiple trigger dengan waktu aksi dan event sama per tabel. Misalkan di tabel A sudah didefinisikan trigger AFTER INSERT, maka kita tidak boleh mendefinisikan trigger AFTER INSERT lagi; namun AFTER EDIT, AFTER DELETE, atau BEFORE (INSERT, EDIT, dan DELETE) bisa diterima.

### D. LATIHAN

Dalam latihan ini digunakan dua buah tabel bernama barang dan pembelian dengan data sebagai seperti berikut:

Tabel barang

id_brg	nama_brg	stok
A10	Mouse	10
A11	Keyboard	15
A12	DVD R-W	10

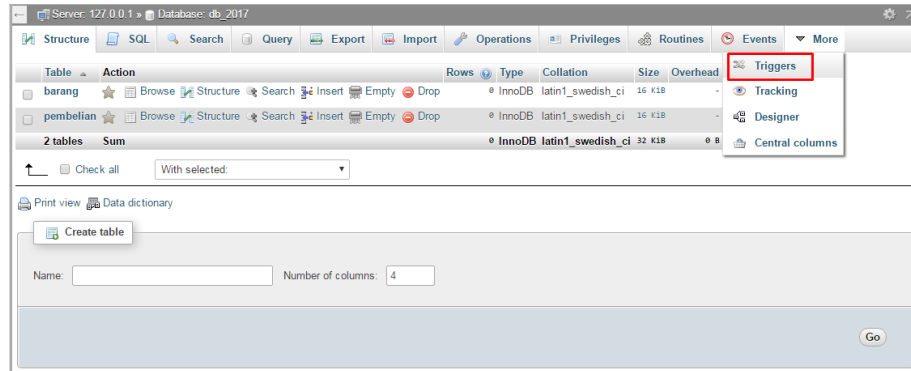
Tabel pembelian

id_pem	id_brg	jml_beli
1	A10	5

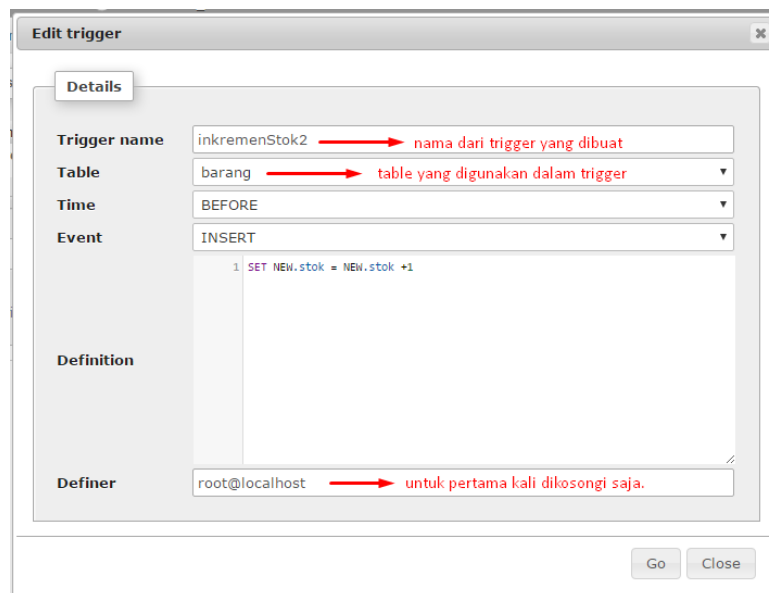
#### 1. Menggunakan Trigger

Operasi-operasi berkenaan dengan pendefinisian trigger tidak berbeda dengan objek-objek database lainnya.

- a. Masuk kedalam database yang digunakan untuk menciptakan tabel tadi, kemudian masuk ke menu *Trigger*.



- b. Kemudian pilih **Add Trigger** pada bagian New. Dan isikan sesuai dengan prosedur berikut.



- c. Setelah berhasil membuat trigger. Kembali ke SQL database dan tuliskan sintak berikut:

***INSERT INTO barang VALUES('A13', 'Modem', 5);***

- d. Dan hasilnya akan seperti berikut.

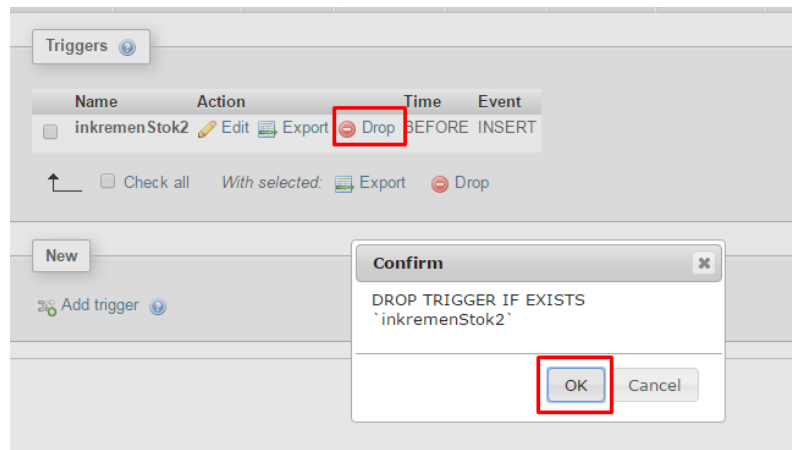


Terlihat bahwa trigger sudah bekerja seperti yang diharapkan, di mana setiap penambahan baru akan menginkremen nilai stok.

Untuk mendapatkan informasi mengenai daftar trigger yang telah terdefinisi, gunakan perintah `SHOW TRIGGERS` atau bisa langsung ke menu trigger

- e. Sebagaimana objek-objek database lainnya, kita menghapus trigger dengan menggunakan perintah `DROP` dengan ketentuan `DROP TRIGGER`

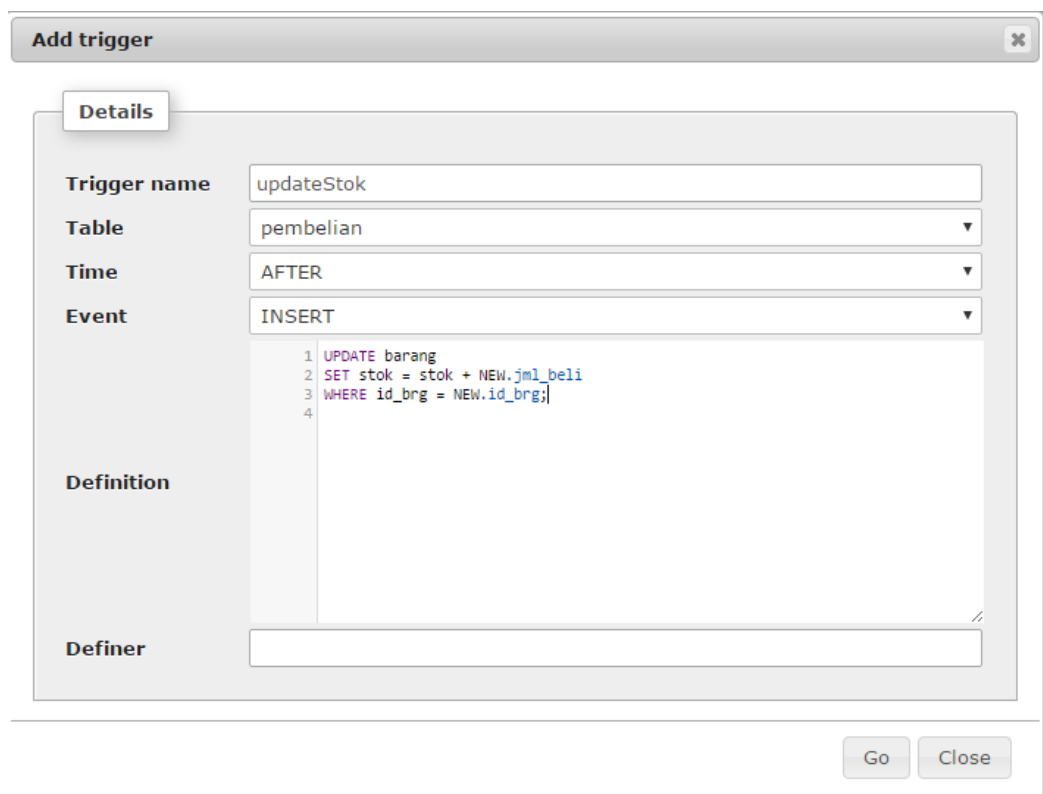
*nama\_trigger*. Atau bisa langsung ke menu trigger dan menghapusnya dengan cara berikut.



## 2. Keyword OLD dan NEW

Untuk merujuk ke kolom-kolom tabel yang diasosiasikan dengan trigger, kita menggunakan keyword OLD dan NEW. Keyword OLD mengacu pada nilai lama, sedangkan NEW merepresentasikan nilai baru.

Di trigger INSERT, kita hanya dapat menggunakan keyword NEW karena tidak ada data lama.

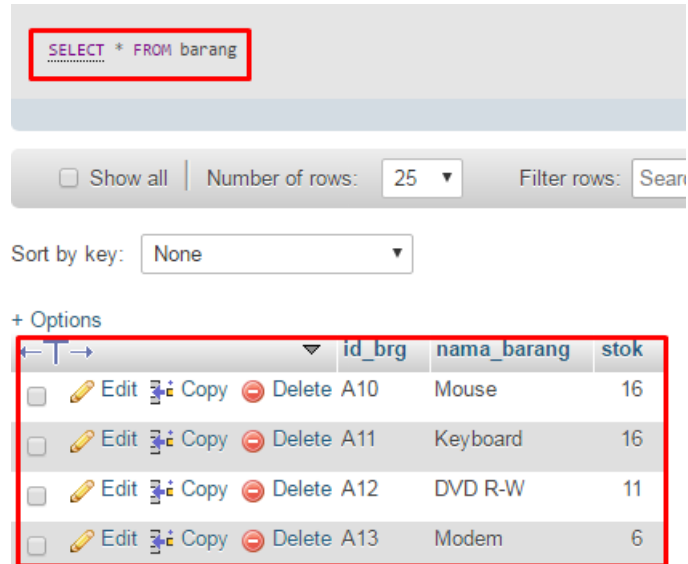


## Praktikum Basis Data 2017 – TE UM

Pada contoh di atas, penambahan data pembelian akan mengakibatkan nilai stok barang berubah menyesuaikan banyaknya nilai jumlah pembelian.

- a. Cek data table barang

***SELECT \* FROM barang***



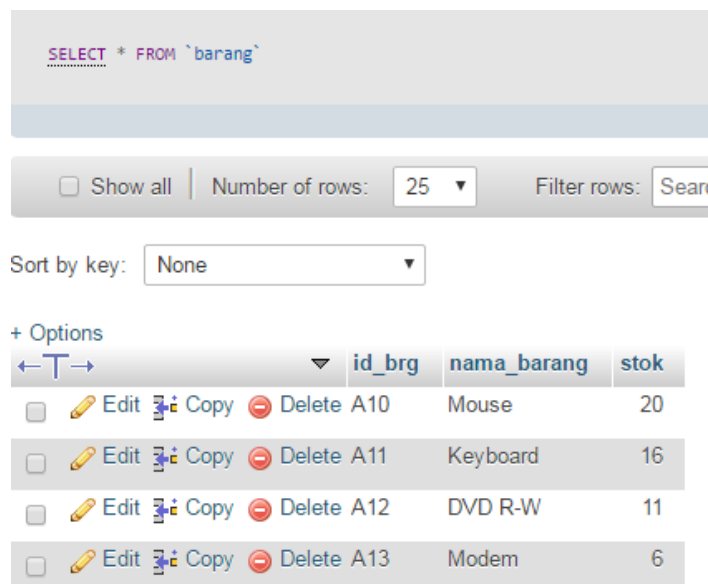
The screenshot shows a database management interface. At the top, the SQL query `SELECT * FROM barang` is entered in a text box. Below the query, there are controls for displaying the results: "Show all" (checked), "Number of rows: 25", and "Filter rows: Search". A "Sort by key: None" dropdown is also visible. The results are displayed in a table with the following data:

				id_brg	nama_barang	stok
<input type="checkbox"/>	Edit	Copy	Delete	A10	Mouse	16
<input type="checkbox"/>	Edit	Copy	Delete	A11	Keyboard	16
<input type="checkbox"/>	Edit	Copy	Delete	A12	DVD R-W	11
<input type="checkbox"/>	Edit	Copy	Delete	A13	Modem	6

- b. Kemudian masukkan data kedalam table pembelian

```
INSERT INTO `pembelian` (`id_pembelian`, `id_brg`, `jml_beli`) VALUES ('2', 'A10', '10');
```

- c. Cek kembali data dalam table barang



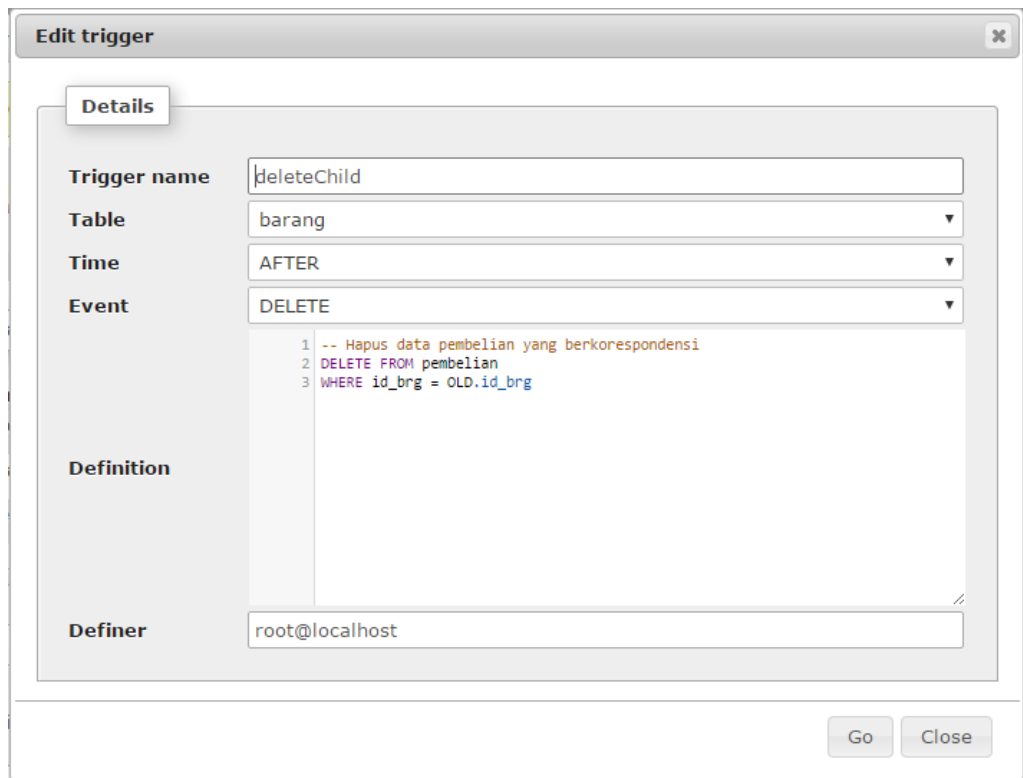
The screenshot shows the same database management interface. The SQL query is now `SELECT * FROM `barang``. The "Number of rows" is still 25. The resulting table data is as follows:

					id_brg	nama_barang	stok
<input type="checkbox"/>	Edit	Copy	Delete	A10	Mouse	20	
<input type="checkbox"/>	Edit	Copy	Delete	A11	Keyboard	16	
<input type="checkbox"/>	Edit	Copy	Delete	A12	DVD R-W	11	
<input type="checkbox"/>	Edit	Copy	Delete	A13	Modem	6	

## Praktikum Basis Data 2017 – TE UM

Untuk kasus trigger DELETE, keyword yang bisa digunakan hanya OLD. Misalkan kita ingin mendefinisikan trigger untuk menghapus semua data pembelian manakala data barang yang sesuai—diindikasikan melalui primary key dan foreign key—dihapus.

- a. Sama seperti langkah sebelumnya. Masuk ke menu trigger



The screenshot shows a window titled "Edit trigger" with a "Details" tab. The fields are as follows:

- Trigger name: deleteChild
- Table: barang
- Time: AFTER
- Event: DELETE
- Definition:

```
1 -- Hapus data pembelian yang berkorespondensi
2 DELETE FROM pembelian
3 WHERE id_brg = OLD.id_brg
```
- Definer: root@localhost

Buttons "Go" and "Close" are visible at the bottom right.

- b. Setelah itu, masuk ke table barang untuk menghapus data.

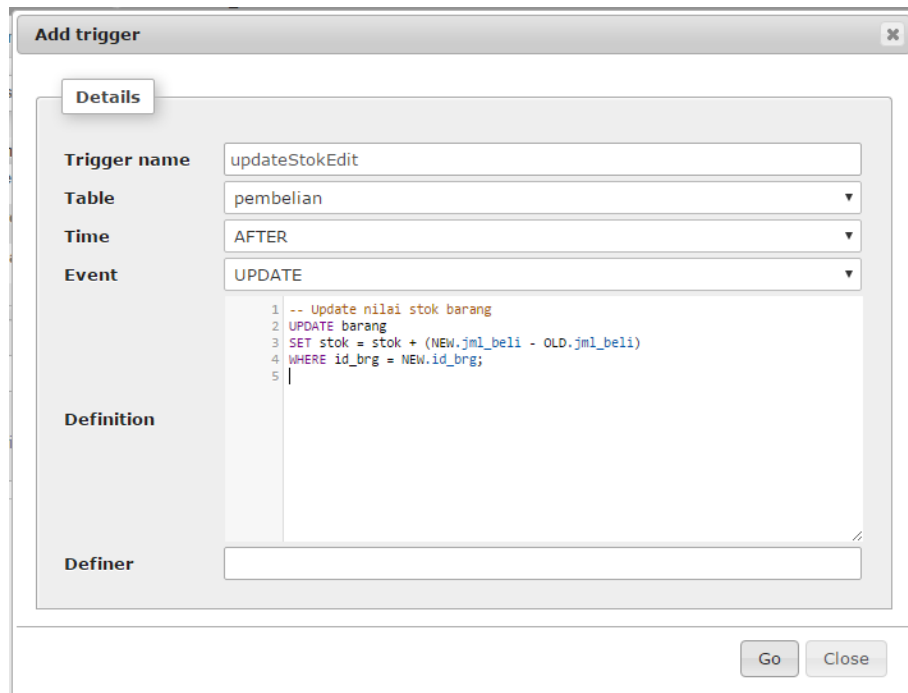
```
DELETE FROM barang WHERE id_brg = 'A10'
```

- c. Kemudian cek data pada table pembelian

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0010 seconds.)

```
SELECT * FROM `pembelian`
```

Khusus untuk operasi UPDATE, kita bisa memanfaatkan keyword NEW maupun OLD.



a. Update pembelian

```
UPDATE pembelian SET jml_beli= 20 WHERE id_pembelian=3
```

b. Tampilkan data pada table barang

```
SELECT * FROM `barang`
```

Show all | Number of rows: 25 | Filter rows: Search

Sort by key: None

+ Options

	id_brg	nama_barang	stok
<input type="checkbox"/> Edit Copy Delete	A10	Mouse	30
<input type="checkbox"/> Edit Copy Delete	A11	Keyboard	16
<input type="checkbox"/> Edit Copy Delete	A12	DVD R-W	11
<input type="checkbox"/> Edit Copy Delete	A13	Modem	6

3. Trigger Kompleks

Keberadaan trigger secara nyata mampu mengatasi berbagai persoalan pelik, misalnya berkaitan dengan integritas atau audit data. Ini tentu sangat masuk akal, karena trigger juga bisa mengandung pernyataan- pernyataan yang kompleks termasuk pencabangan, pengulangan, fungsi-fungsi agregat, bahkan kode pemanggilan prosedur. Sebagai ilustrasi sederhana, kita bisa mendefinisikan trigger untuk memeriksa operasi penambahan data barang. Skenarionya, jika data sudah ada, berikan status eksistensi barang; sebaliknya, data bisa langsung dimasukkan.

```
CREATE TRIGGER auditBarang
  BEFORE INSERT ON barang

  FOR EACH ROW BEGIN

    IF NOT EXISTS (SELECT id brg FROM barang WHERE id brg =
NEW.id_brg)
  THEN
    SET NEW.nama_brg = NEW.nama_brg, NEW.stok = NEW.stok;
  ELSE
    SET @status = CONCAT('Id ', NEW.id_brg, ' sudah ada');
  END IF;
```



E. PRAKTIKUM

1. Modifikasi trigger INSERT pembelian untuk menambahkan fitur bonus di dalamnya. Aturannya adalah jika pembelian > 50 dan < 100, maka bonus = 5; jika pembelian > 100 dan < 150, maka bonus = 10; jika pembelian > 150, maka bonus = 20. Ingat, aturan penyesuaian stok barang masih berlaku, setiap ada pembelian maka stok akan berkurang.

a. Setting stok

+ Options				id_brg	nama_barang	stok
<input type="checkbox"/>				A10	Mouse	100
<input type="checkbox"/>				A11	Keyboard	200
<input type="checkbox"/>				A12	DVD R-W	300
<input type="checkbox"/>				A13	Modem	200

b. Tambahkan pembelian













Server: 127.0.0.1 » Database: promo » Table: pembelian

Column	Type	Function	Null	Value
id_pem	int(10)	<input type="text"/>	<input type="checkbox"/>	1
id_brg	varchar(150)	<input type="text"/>	<input type="checkbox"/>	A10
jml_beli	int(10)	<input type="text"/>	<input type="checkbox"/>	60

c. Pembelian akan diberikan bonus

+ Options				id_pem	id_brg	jml_beli
<input type="checkbox"/>				1	A10	65

d. Stok akan otomatis menyesuaikan sesuai aturan

+ Options				id_brg	nama_barang	stok
<input type="checkbox"/>	 Edit	 Copy	 Delete	A10	Mouse	35
<input type="checkbox"/>	 Edit	 Copy	 Delete	A11	Keyboard	200
<input type="checkbox"/>	 Edit	 Copy	 Delete	A12	DVD R-W	300
<input type="checkbox"/>	 Edit	 Copy	 Delete	A13	Modem	200

F. TUGAS RUMAH

1. Buat tabel **pembayaran**

id_pem	Int(10), Primary
jumlah_pem	Int(15)

**Tambahkan** filed “**Harga**” pada table Barang, lalu modifikasi Trigger INSERT pada table pembelian untuk menghitung nota pembayaran, contoh jika satuan harga barang A = 1000, dan terjadi pembelian sebanyak 10 item barang, maka jumlah\_pem akan tercatat 10000. Ingat, aturan bonus dan penyesuaian stok barang masih berlaku.

2. Buat tabel **log\_pembelian**

operasi	Varchar(25), Primary
waktu	date

Selanjutnya buatlah trigger di dalam tabel **Pembelian** untuk merekam operasi INSERT, UPDATE, DELETE, dan kemudian menyimpan rekaman operasi sebagai bukti transaksi di tabel log\_pembelian.

- Jika INSERT, maka akan memasukkan operasi “Insert” dan menampilkan waktu operasinya di tabel log\_pembelian
- Jika UPDATE, maka akan memasukkan operasi “Update” dan menampilkan waktu operasinya di tabel log\_pembelian
- Jika DELETE, maka akan memasukkan operasi “Delete” dan menampilkan waktu operasinya di tabel log\_pembelian

*Note :*

*Jika terjadi error duplikasi pendefinisian trigger, terlebih dahulu simpan query trigger di dalam notepad lalu hapus semua trigger di tabel pembelian*