1. TUJUAN

MODUL 8 TRIGGER

* 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.

1. 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
2. DASAR TEORI

# 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 lingkupnya, 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.

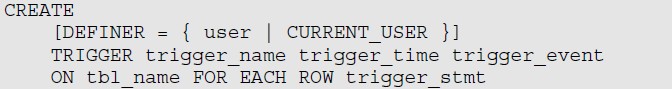
Dari sisi perilaku (behavior) eksekusi, trigger dapat dibedakan menjadi beberapa jenis; namun umumnya ada dua jenis: trigger BEFORE dan AFTER. Sesuai penamaannya, jenis-jenis ini merepresentasikan waktu eksekusi trigger— misalnya sebelum ataukah sesudah pernyataan- pernyataan yang berkorespondensi. Adakalanya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

# 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.

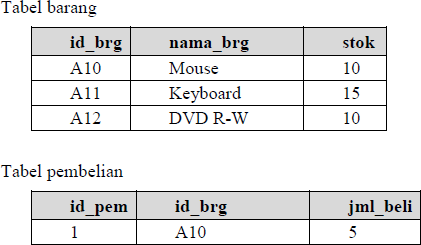
Sintaks pendefinisian trigger diperlihatkan sebagai berikut:



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.

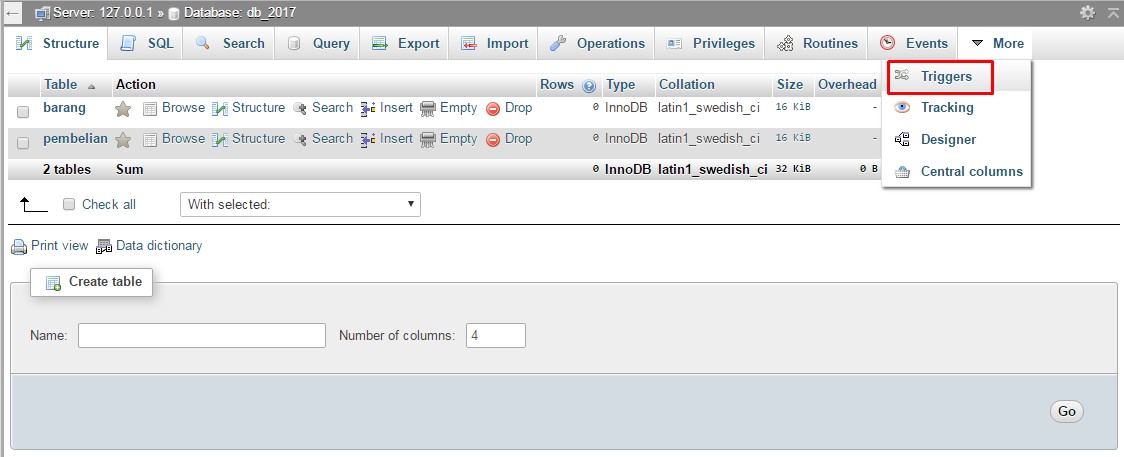
1. LATIHAN

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

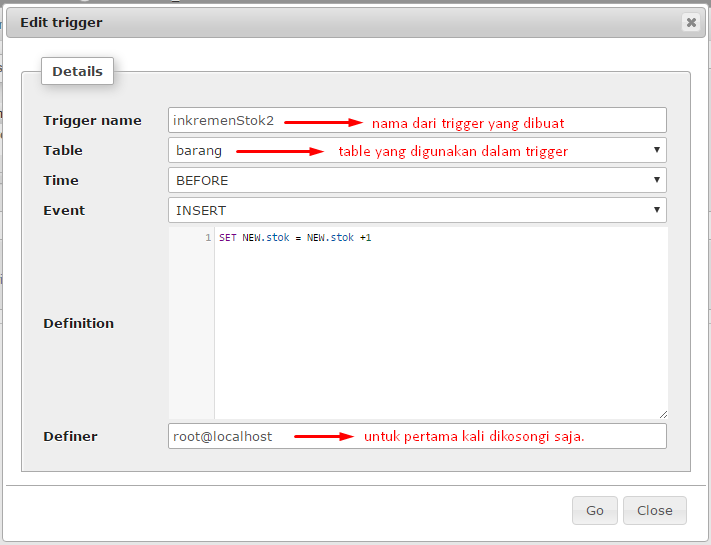
1. Menggunakan Trigger

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

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



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



* 1. Setelah berhasil membuat trigger. Kembali ke SQL database dan tuliskan sintak berikut:

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

* 1. Dan hasilnya akan seperti berikut.

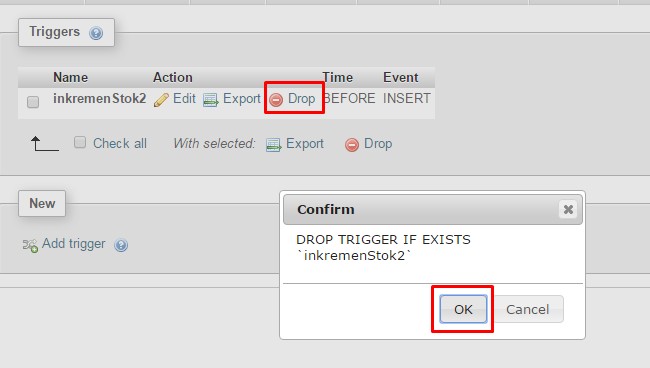


Terlihat bahwa trigger sudah bekerja seperti yang diharapkan, di mana setiap penambahan baru akan menginkremen nilai stok.

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

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

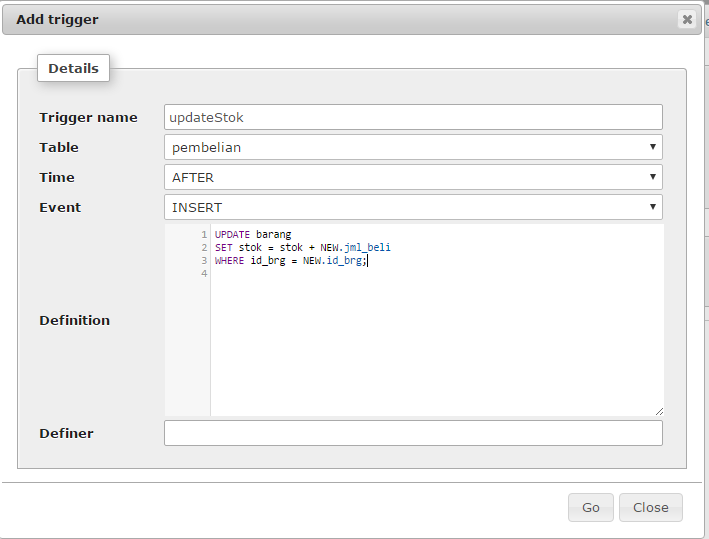
*nama\_trigger.* Atau bisa langsung ke menu trigger dan menghapusnya dengan cara berikut.



1. 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.

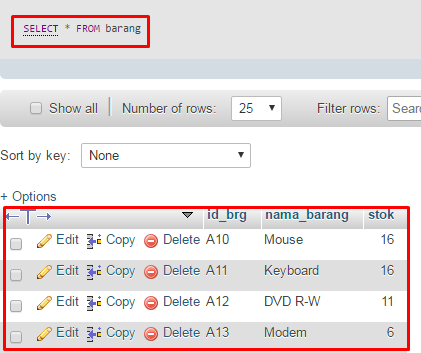
Di trigger INSERT, kita hanya dapat menggunakan keyword NEW karena tidak ada data lama.



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

* 1. Cek data table barang

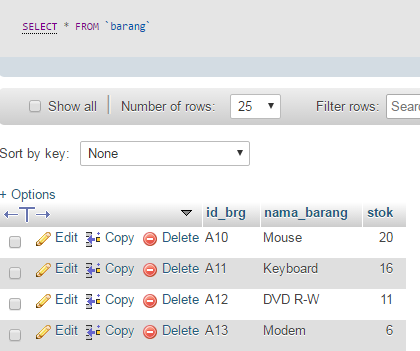
## SELECT \* FROM barang



* 1. Kemudian masukkan data kedalam table pembelian

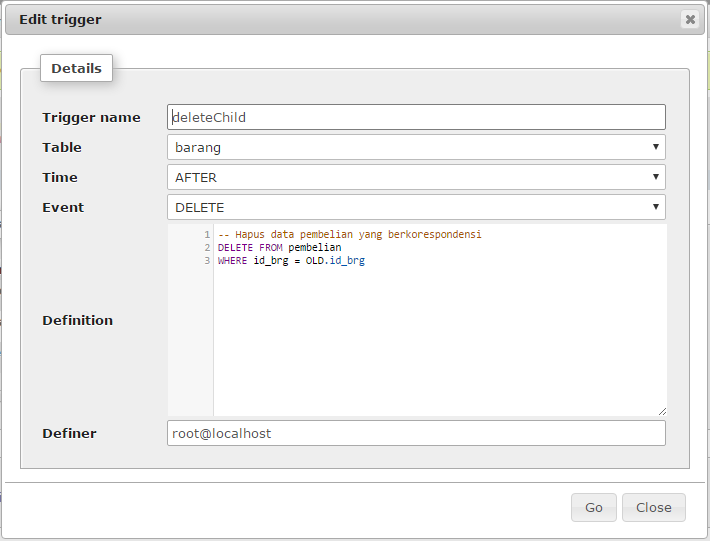


* 1. Cek kembali data dalam table barang



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.

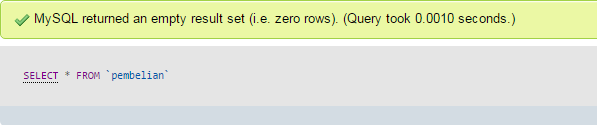
1. Sama seperti langkah sebelumnya. Masuk ke menu trigger



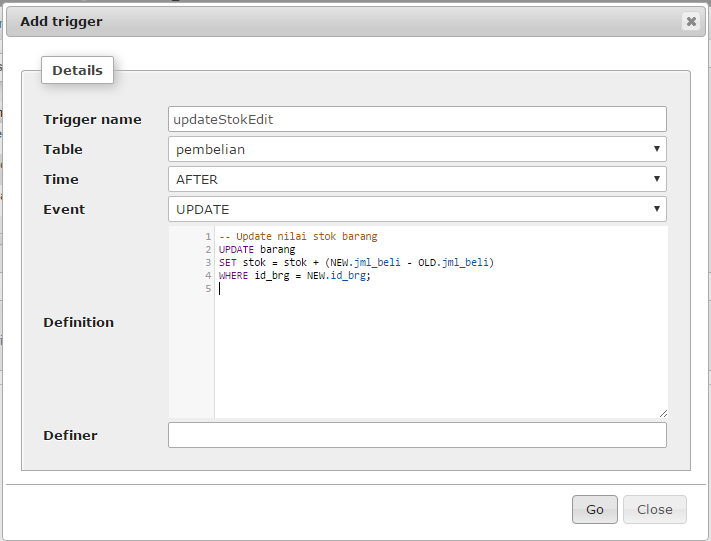
1. Setelah itu, masuk ke table barang untuk menghapus data.



1. Kemudian cek data pada table pembelian



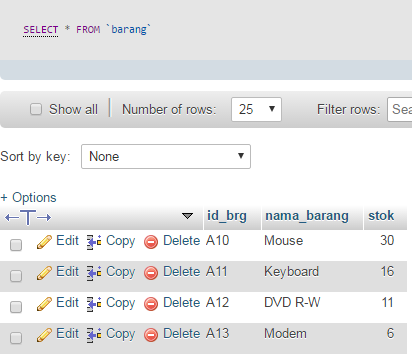
Khusus untuk operasi UPDATE, kita bisa memanfaatkan keyword NEW maupun OLD.



1. Update pembelian

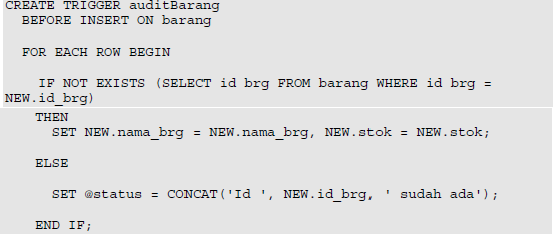


1. Tampilkan data pada table barang

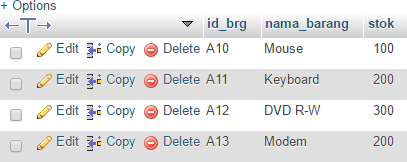


1. Trigger Kompleks

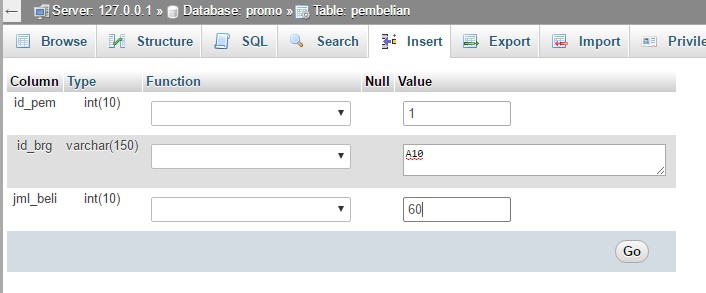
Keberadaan trigger secara nyata mampu mengatasi berbagai persoalan pelik, misalnya berkaitan dengan integritas atau audit data. Ini tentu sangat masuk akal, karena trigger juga bisa mengandung pernyataan- pernyataan yang kompleks termasuk pencabangan, pengulangan, fungsi-fungsi agregat, bahkan kode pemanggilan 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.



1. PRAKTIKUM
2. 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.
   1. Setting stok



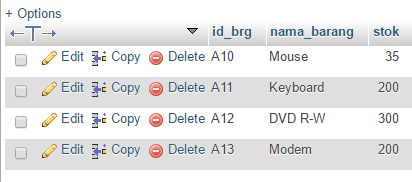
* 1. Tambahkan pembelian



* 1. Pembelian akan diberikan bonus



* 1. Stok akan automatis menyesuaikan sesuai aturan



1. TUGAS RUMAH
2. 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.

1. 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*