MODUL 10 – TRIGGER

1.1. CAPAIAN PEMBELAJARAN

1. Mahasiswa mampu menjelaskan trigger dalam SQL
2. Mahasiswa mampu mengimplementasikan trigger dalam SQL

1.2. ALAT DAN BAHAN

1. Seperangkat komputer lengkap/Laptop dengan koneksi internet
2. Sistem Operasi Windows/Mac/Linux
3. Aplikasi Paket Web server XAMPP
4. Aplikasi Kantor (Microsoft Office/Libre Office/WPS Office/etc)

4.1. 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 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.

2. Trigger dalam 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.

4.2. PRAKTIKUM

Dalam praktikum 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

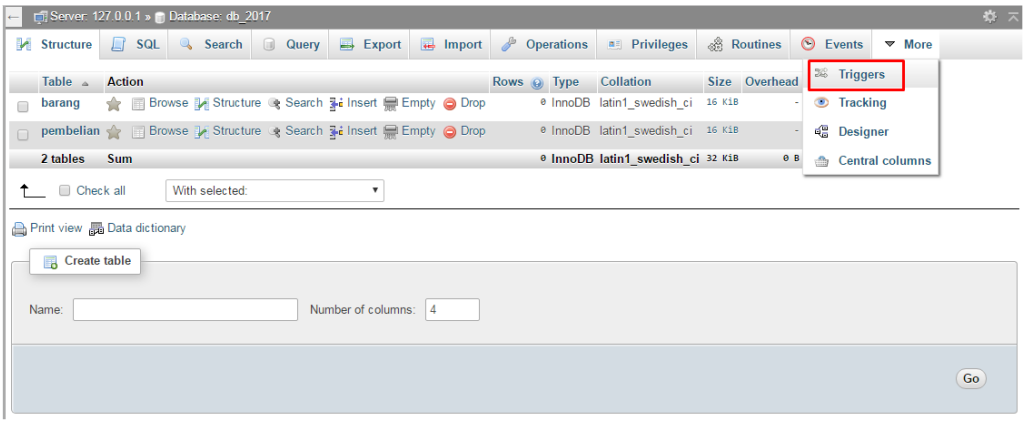
Tabel pembelian

id_pem	id_brg	jml_beli
1	A10	5

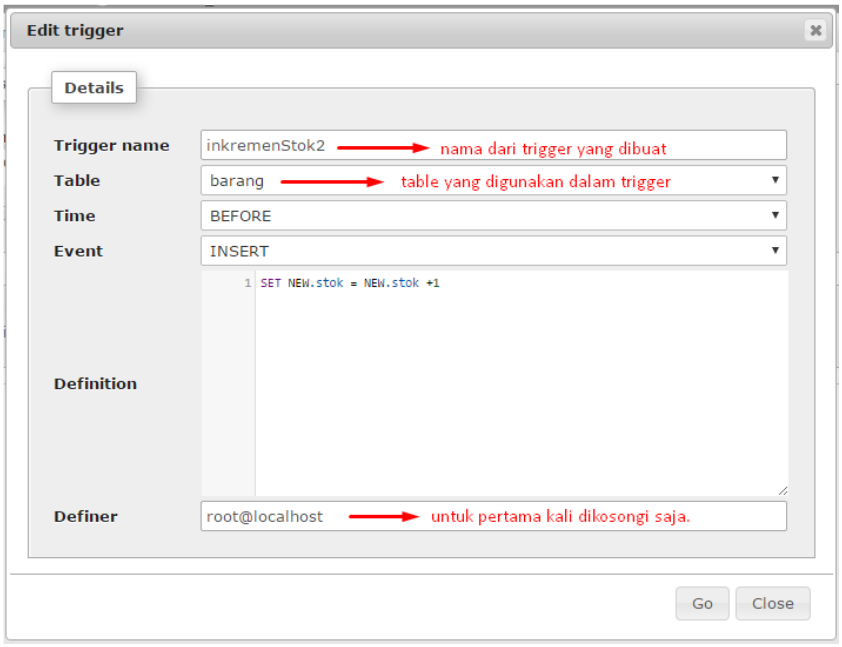
1. Menggunakan trigger

   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. Pilih Add Trigger pada bagian New. Dan isikan sesuai dengan gambar berikut.

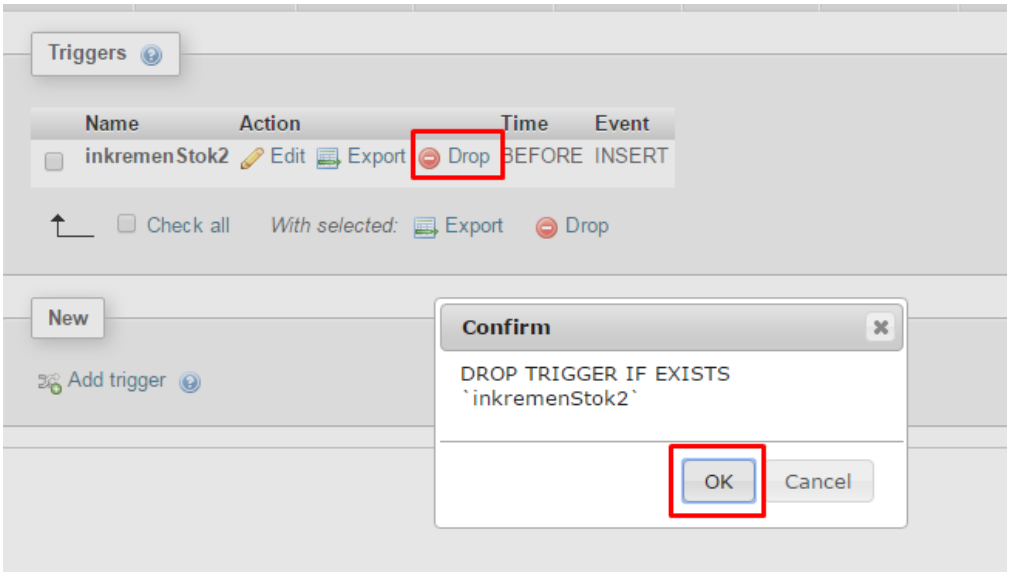


c. Setelah berhasil membuat trigger. Kembali ke SQL database dan tuliskan sintak berikut:

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

d. Jika berhasil maka akan menampilkan data seperti pada gambar berikut

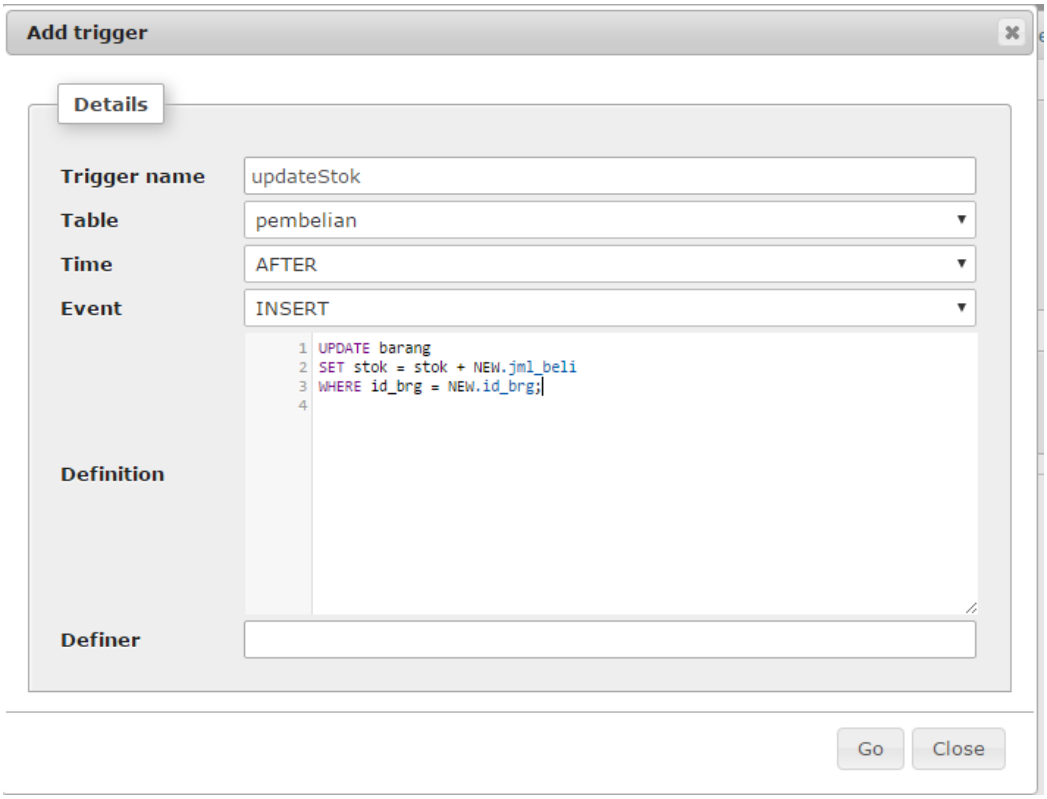
e. Sebagaimana objek-objek database lainnya, kita menghapus trigger dengan menggunakan perintah DROP dengan ketentuan DROP TRIGGER nama_trigger. Atau bisa langsung ke menu trigger dan menghapusnya dengan cara berikut.



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.



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

a. Cek tabel barang

SELECT * FROM barang

```
SELECT * FROM barang
```

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

Sort by key: None

+ Options

	id_brg	nama_barang	stok
Edit Copy Delete	A10	Mouse	16
Edit Copy Delete	A11	Keyboard	16
Edit Copy Delete	A12	DVD R-W	11
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

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

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

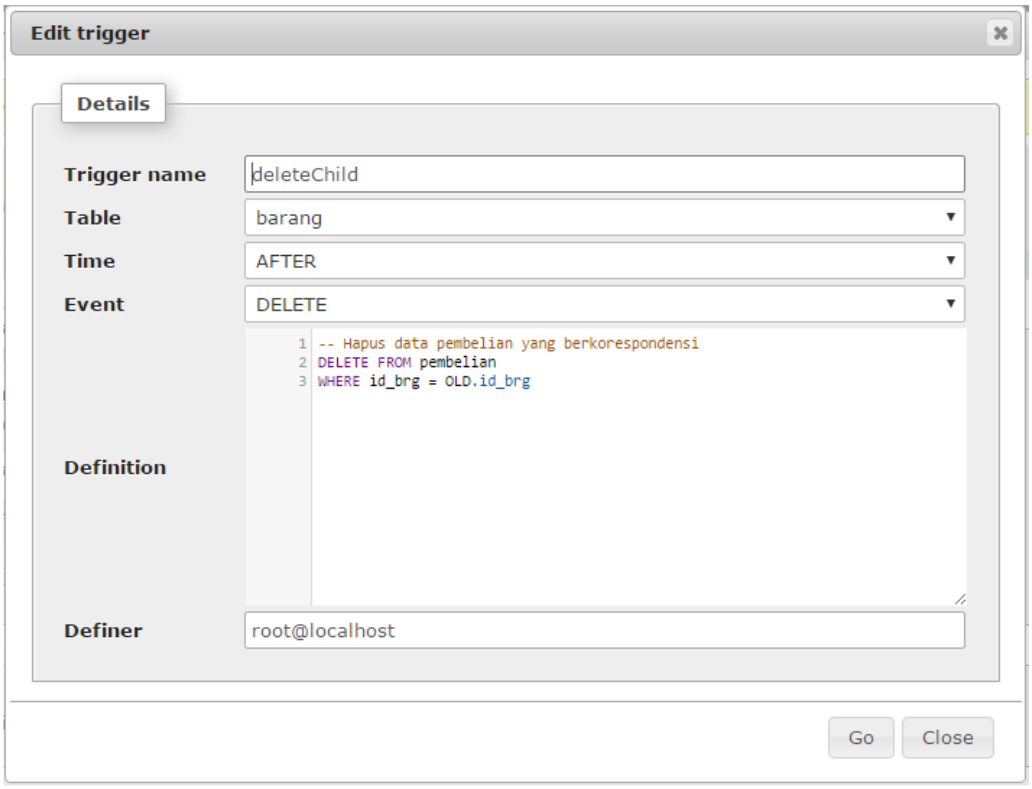
Sort by key: None

+ Options

	id_brg	nama_barang	stok
Edit Copy Delete	A10	Mouse	20
Edit Copy Delete	A11	Keyboard	16
Edit Copy Delete	A12	DVD R-W	11
Edit Copy Delete	A13	Modem	6

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.

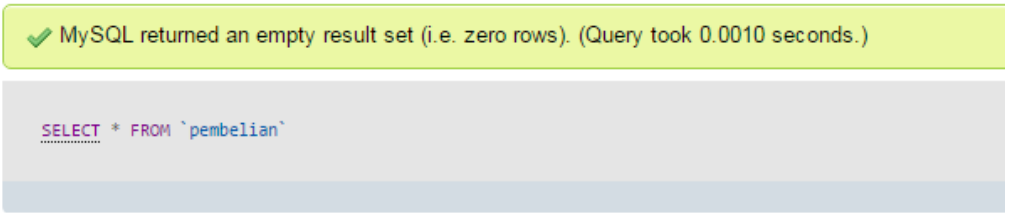
a. Membuat trigger baru seperti langkah sebelumnya



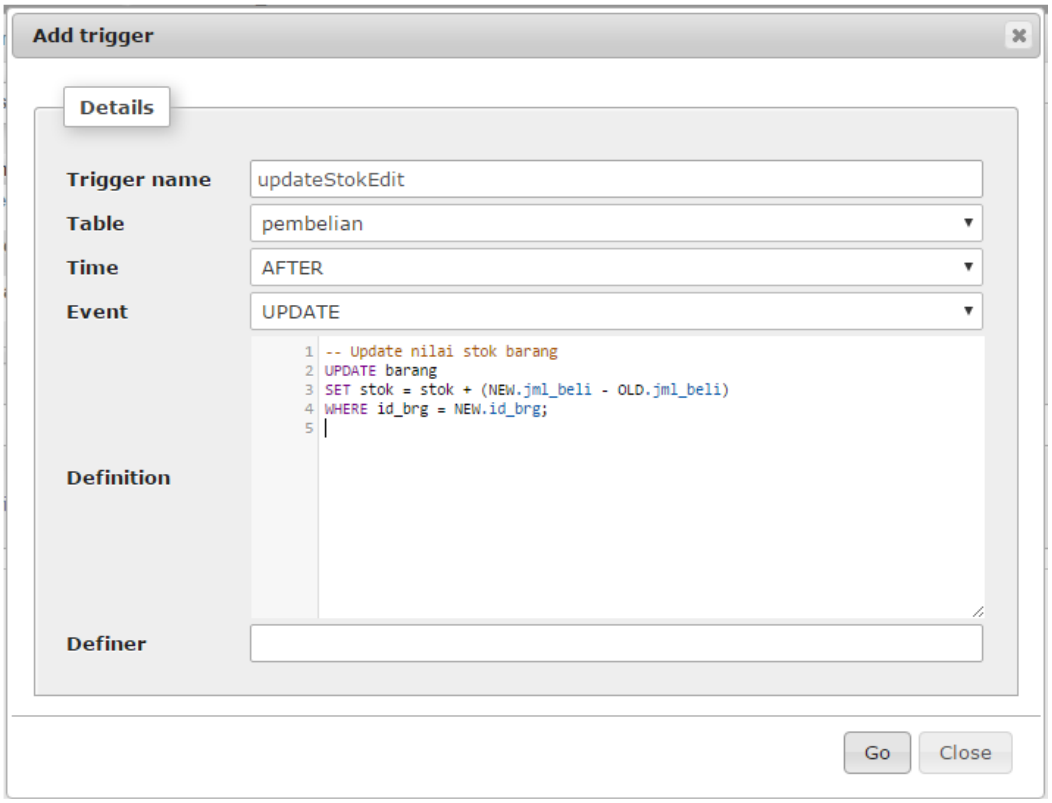
b. Hapus data pada tabel barang

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

c. Cek apakah trigger berjalan dengan baik dengan menampilkan tabel pembelian



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



a. Lakukan update pada tabel pembelian

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

b. Cek apakah trigger berjalan dengan baik dengan cara tampilkan pada tabel barang

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

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

Sort by key: None

+ Options

	id_brg	nama_barang	stok
Edit Copy Delete	A10	Mouse	30
Edit Copy Delete	A11	Keyboard	16
Edit Copy Delete	A12	DVD R-W	11
Edit Copy Delete	A13	Modem	6

3. Trigger Kompleks

Keberadaan trigger secara nyata mampu mengatasi berbagai persoalan sulit, 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;
```

4.3. TUGAS MODUL 10

4.3.1. Soal

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.

4.3.2. Petunjuk Pengerjaan

a) Tugas:

- Tugas membuat perintah SQL sesuai pada soal
- Tuliskan perintah dan screenshoot hasil perintah.

b) Laporan:

- Buatlah laporan akhir berdasarkan diagram yang Anda buat.
- Laporan dibuat sesuai format dan **diketik**.
- Masukkan langkah-langkah pengerjaan tugas ke dalam laporan dalam bentuk screenshoot dan penjelasan.
- **Cantumkan tanda tangan** Anda di setiap halaman dokumen laporan.
- Jika di dalam laporan ada gambar atau *screenshot* yang ingin ditampilkan, bisa

ditempelkan pada halaman(menyesuaikan)

- Laporan disimpan dalam bentuk pdf
- Penamaan *file* pdf : " **LaporanModul10_DBMS_NPM.pdf**"

c) Pengumpulan:

- *File* yang dikumpulkan yaitu:
    - Laporan : **TugasModul10_DBMS_NPM.pdf**
- **Batas Pengumpulan**: Sebelum Pertemuan Praktik Ke 14.