MATERI VII
TRIGGER

A. Tujuan

1. Mahasiswa dapat menjelaskan struktur sintaks dan komponen di dalam pembuatan *trigger*.
2. Mahasiswa dapat mengimplementasikan pembuatan *trigger* dalam berbagai operasi *insert*, *update*, dan *delete*.
3. Mahasiswa dapat mengimplementasikan pemanggilan *stored function* di dalam *trigger*.

B. Dasar Teori

Sebuah basis data dapat memproses permintaan perubahan datanya hingga ribuan data per detiknya. Permintaan perubahan data meliputi perintah-perintah *insert, edit, alter,* atau *delete*. Terkait perubahan data, dibutuhkan manajemen data yang bagus sehingga dapat mempertahankan konsistensi pada data yang saling memiliki hubungan secara logika dengan melakukan validasi sebelum perubahan tersebut disimpan. Sebelum MySql versi 5.0.2, Mysql hanya dapat melakukan validasi data meliputi pembatasan masukan sesuai tipe data pada kolom, penggunaan *foreign key* untuk memastikan *referential integrity*. Sejak versi MySQL 5.0.2, validasi perubahan data yang dilakukan oleh perintah-perintah *insert, edit, alter,* atau *delete* dapat dilakukan menggunakan *trigger*.

Trigger adalah kumpulan statemen yang disimpan dan dijalankan ketika suatu event terjadi pada suatu kolom atau tabel tertentu. Umumnya *trigger* digunakan untuk menjalankan statemen query secara otomatis pada kondisi sebelum atau sesudah proses INSERT, UPDATE atau DELETE dari suatu tabel. Sebagai contoh, misalnya untuk menyimpan id pelanggan secara otomatis ke tabel “log” sebelum menghapus data di tabel pelanggan, melakukan update data otomatis jika terjadi perubahan dalam sistem penjualan, jika dientri barang baru maka stock akan bertambah secara otomatis, atau menyimpan setiap aktivitas pada basis data ke dalam tabel “log”.

Trigger dapat mengakses record tabel sebelum atau sesudah proses dengan menggunakan *NEW* dan *OLD*. *NEW* digunakan untuk mengambil record yang akan

diproses (insert atau update), sedangkan *OLD* digunakan untuk mengakses record yang sudah diproses (*update* atau *delete*).

C. Sintak *Trigger*

Komponen yang harus ada di dalam *trigger* adalah 1) nama trigger, 2) *activation time*, 3) kondisi aktifasi *trigger,* 4) tabel yang akan mengaktifkan *trigger* sesuai kondisi *trigger* yang diberikan, dan 5) aksi yang dilakukan oleh *trigger*. Sintaks *trigger* secara umum ditunjukkan pada gambar 1 berikut :

```
CREATE TRIGGER name
[BEFORE|AFTER] [INSERT|UPDATE|DELETE]
ON tablename
FOR EACH ROW statement
```

Gambar 1. Sintak *Trigger*

Activation time pada *trigger* adalah *before* dan *after. Before* berarti *trigger* akan dijalankan sebelum proses INSERT, UPDATE atau DELETE, sedangkan *after* berarti *trigger* akan dijalankan setelah proses INSERT, UPDATE atau DELETE. Sebagai contoh *trigger* untuk menyimpan proses INSERT yang terjadi pada tabel mahasiswa ke dalam tabel "log". Sebelumnya buat terlebih dahulu table dengan nama "tb_log" dengan sintak seperti ditunjukkan pada gambar 2.

```
CREATE TABLE `tb_log` (
        `log_id` int(8) NOT NULL AUTO_INCREMENT,
        `Nim` Varchar(12) NOT NULL,
        `perubahan` enum('CREATE','UPDATE','DELETE') NOT NULL,
        `waktu` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
        PRIMARY KEY (`log_id`)
)
```

Gambar 2. Sintak Table "tb_log"

Kemudian buat *trigger* untuk menangani contoh di atas. Ketika ada proses penambahan data di tabel "mahasiswa" akan menambahkan record secara otomatis ke dalam table "tb_log". Sintak *trigger* ditunjukkan pada gambar 3.

```
DELIMITER $$
CREATE TRIGGER `after_insert_mahasiswa`
    AFTER INSERT ON `mahasiswa` FOR EACH ROW
    BEGIN
                        SET @tipe = 'CREATE';
      INSERT INTO tb_log (Nim, perubahan,waktu) VALUES (NEW.Nim, @tipe,now());
    END;
$$
```

Gambar 3. Sintak *Trigger Insert* Data Mahasiswa

Setelah *trigger* seperti gambar 3 sukses dibuat. Tambahkan *record* baru pada table "mahasiswa. Kemudian buka table "tb_log', apakah ada *record* baru yang ditambahkan sebagai efek dari penambahan *record* pada tabel "mahasiswa".

D. *Trigger* untuk Update Data

Selain untuk menangani ketika terjadi proses penambahan data pada suatu table, *trigger* juga dapat digunakan untuk menangani proses perubahan data yang terkait dengan data lainnya. Sebagai contoh, ketika suatu data NIM di table mahasiswa diubah, maka data NIM tersebut di table "krs" juga akan mengalami perubahan untuk menjaga konsistensi data. Sintak *trigger* untuk menangani perubahan data ditunjukkan pada gambar 4.

```
DELIMITER $$
CREATE TRIGGER `after_update_mahasiswa`
    AFTER UPDATE ON `mahasiswa` FOR EACH ROW
    BEGIN
        SET @tipe = 'UPDATE';
        UPDATE krs set NIM = NEW.NIM;
        INSERT INTO tb_log (Nim, perubahan,waktu) VALUES (NEW.Nim, @tipe,now());
    END;
$$
```

Gambar 4. Sintak *Trigger Update* Data Mahasiswa

Ubahlah salah satu data "NIM" pada tabel "mahasiswa", kemudian buka tabel "krs", pastikan perubahan NIM pada tabel "mahasiswa" berpengaruh pada tabel "krs". Selanjutnya buka table "tb_log", pastikan terdapat satu *record* baru pada tabel "tb_log" dengan nilai pada *field* "perubahan" yaitu "UPDATE".

E. *Trigger* dengan Pemanggilan *Stored Function*

Selain perintah INSERT, UPDATE, atau DELETE, *trigger* dapat digunakan untuk memanggil *stored function* untuk mendapatkan nilai kembalian yang dihasilkan oleh *stored function* kemudian menyimpan di dalam variable yang dideklarasikan di dalam *trigger*. Sebagai contoh, *trigger* untuk menghasilkan NIM baru melalui sebuah *stored function* sebelum proses *insert* data mahasiswa baru ke dalam tabel "mahasiswa". Sintaks *trigger* yang melibatkan pemanggilan *stored function* ditunjukkan pada gambar 5.

```
DELIMITER $$
CREATE TRIGGER buat_NIM
        BEFORE INSERT ON mahasiswa
        FOR EACH ROW
        BEGIN
                        DECLARE s VARCHAR(12);
                        SET s = (SELECT NIM_Baru());
                        SET @tipe = 'CREATE';

                        IF(NEW.Nim IS NULL OR NEW.Nim = '')THEN
                                SET NEW.Nim =s;
                                INSERT INTO tb_log (Nim, perubahan,waktu) VALUES (NEW.Nim, @tipe,now());
                        END IF;
        END;
$$
```

Gambar 5. Pemanggilan *Stored Function* Di Dalam *Trigger*

Pemanggilan *stored function* melalui *trigger* pada gambar 5 dilakukan pada baris sintaks "*SET s = (SELECT NIM_baru());*", nilai kembalian yag dihasilkan oleh *stored function* disimpan ke dalam variable "s". Selanjutnya, ketika terjadi proses penambahan *record* baru pada tabel mahasiswa, nilai *field* Nim akan diisi oleh nilai yang disimpan pada variable "s".

F. Evaluasi dan Pertanyaan

1. Buatlah *stored function* untuk menghasilkan penomoran Nip secara otomatis sesuai berdasarkan Nip terakhir pada tabel "dosen". Selanjutnya buatlah *trigger* untuk menambahkan data dosen baru dengan penomoran Nip merupakan hasil dari *stored function*. Buat juga statemen query pada *trigger* tersebut untuk menambahkan *record* ke dalam tabel "tb_log".
2. Buatlah *trigger* untuk proses *update* kode mata kuliah pada tabel "jadwal" dan "krs" ketika ada perubahan kode mata kuliah pada tabel "mata_kuliah". Buat juga statemen query pada *trigger* tersebut untuk menambahkan *record* ke dalam tabel "tb_log" ketika terjadi perubahan kode mata kuliah pada tabel "mata_kuliah", tabel "jadwal", dan tabel "krs".
3. Buatlah *trigger* untuk proses *update* kode Nip pada tabel "jadwal" ketika tejadi perubahan Nip pada tabel "dosen". Buat juga statemen query pada *trigger* tersebut untuk menambahkan *record* ke dalam tabel "tb_log" ketika terjadi perubahan Nip pada tabel "jadwal", dan tabel "dosen".