13

TRIGGER

Oleh : Kurniawan, S.Kom

DEFINISI TRIGGER

- Trigger adalah sebuah objek database yang diasosiasikan dengan sebuah tabel dan akan aktif (terpicu/trigger) ketika sebuah event terjadi pada tabel tersebut.
- Trigger hanya terjadi ketika ada eksekusi INSERT, DELETE dan UPDATE pada table yang bersangkutan.
- Waktu eksekusi trigger yang mungkin terjadi terdiri dari 2 yaitu BEFORE dan AFTER dari statement SQL-nya.

KEUNTUNGAN MENGGUNAKAN TRIGGER

- Trigger dapat digunakan untuk mengubah data sebelum proses INSERT dilakukan atau untuk memberikan nilai default. Misalnya mengubah data yang diluar nilai yang diperbolehkan. Misalnya, jika ada pengisian nilai di atas 100, maka akan dijadikan 100.
- Anda dapat menyimpan data suatu record ke tabel lain (misalnya history) sebelum data tersebut diupdate atau didelete. Sehingga semua perubahan data dapat terlacak dari sejak data itu dibuat.

STRUKTUR TRIGGER

```
CREATE TRIGGER nama_trigger
      { BEFORE | AFTER }
      { INSERT | UPDATE | DELETE }
ON nama_tabel
FOR EACH ROW
      statement-statement
```

MENGAKSES NILAI BARU DAN LAMA

- Di dalam trigger, anda dapat mengakses data lama dan data baru. Data lama dapat direference dengan record OLD dan data baru dapat direference dengan record NEW.

OPERASI	NEW (READ/WRITE)	OLD (READ)
INSERT	√	
UPDATE	√	√
DELETE		√

- Untuk mengacu ke sebuah field dapat ditulis dengan NEW.namafield atau OLD.namafield.

CONTOH 1

- Ada sebuah tabel mahasiswa {nim, nama, alamat}.
- Buatlah sebuah trigger yang akan menyimpan history alamat. Jika sebuah alamat berubah, maka alamat lama harus disimpan ke tabel history alamat.

CONTOH 1

```
CREATE TABLE mhs(
    nim VARCHAR(8) PRIMARY KEY,
    nama VARCHAR(50) NOT NULL,
    alamat VARCHAR(200) NOT NULL
)
```

```
INSERT INTO mhs VALUES
(10107001,'AA', 'Jl. AA No. 1'),
(10107002,'BEBE', 'Jl. BB No. 2'),
(10107003,'CECE', 'Jl. CC No. 3')
```

```
CREATE TABLE history_alamat_mhs(
    waktu datetime,
    nim VARCHAR(8),
    alamat VARCHAR(200),
    oleh VARCHAR(50)
)
```

CONTOH 1

```
DELIMITER //
DROP TRIGGER IF EXISTS trig_update_mhs //
CREATE TRIGGER trig_update_mhs
AFTER UPDATE ON mhs
FOR EACH ROW
BEGIN
     INSERT INTO history_alamat_mhs
         VALUES(now(),OLD.nim,OLD.alamat,USER());
END //
```

CONTOH 1

- Contoh Penggunaan Trigger

```
UPDATE mhs SET alamat='Jl. Dipati Ukur No. 1'
    WHERE nim=10107001;
UPDATE mhs SET alamat='Jl. Ir. H. Djuanda No. 100'
    WHERE nim=10107002;
UPDATE mhs SET alamat='Jl. Merdeka No. 3'
   WHERE nim=10107001;
```

```
SELECT * FROM history_alamat_mhs;
```

waktu	nim	alamat	oleh
2010-01-02 11:20:52	10107001	Jl. AA No. 1	root@localhost
2010-01-02 11:20:54	10107002	Jl. BB No. 2	root@localhost
2010-01-02 11:20:55	10107001	Jl. Dipati Ukur No. 1	root@localhost

CONTOH 1

- Untuk melihat semua alamat yang pernah digunakan oleh mhs yang ber nim 10107001 adalah

```
(SELECT now() waktu,alamat FROM mhs WHERE nim=10107001)
UNION
(SELECT waktu,alamat FROM history_alamat_mhs WHERE nim=10107001)
ORDER BY waktu DESC
```

waktu	alamat
2010-01-02 12:24:02	Jl. Merdeka No. 3
2010-01-02 11:20:55	Jl. Dipati Ukur No. 1
2010-01-02 11:20:52	Jl. AA No. 1

CONTOH 1

- Trigger pertama mempunyai kekurangan yaitu ketika ada perubahan di tabel mhs walaupun tidak mengubah kolom alamat, maka statement INSERT di tabel history akan dijalankan. Trigger ini bisa dioptimalkan dengan membuat trigger seperti di bawah ini.

```
DELIMITER //
DROP TRIGGER IF EXISTS trig_update_mhs //
CREATE TRIGGER trig_update_mhs
AFTER UPDATE ON mhs
FOR EACH ROW
BEGIN
      IF OLD.alamat<>NEW.alamat THEN
        INSERT INTO history_alamat_mhs
           VALUES(now(),OLD.nim,OLD.alamat,USER());
      END IF;
END //
```

History hanya dicatat kalau ada perubahan di alamat

CONTOH 2

- Buatlah suatu trigger yang akan dieksekusi ketika ada perubahan NIM di tabel mhs yang akan melakukan update ke tabel history_alamat_mhs untuk menyesuaikan NIM-nya agar relasinya tidak terlepas.

```
DELIMITER //
DROP TRIGGER IF EXISTS trig_update_nim //
CREATE TRIGGER trig_update_nim
AFTER UPDATE ON mhs
FOR EACH ROW
BEGIN
      IF OLD.nim<>NEW.nim THEN
        UPDATE mhs SET nim=NEW.nim WHERE nim=OLD.nim;
      END IF;
END //
```

CONTOH 2

- Jika anda mengeksekusi trigger tersebut maka anda akan melihat ada peringatan yang berisi : "*This version of MySQL doesn't yet support 'multiple triggers with the same action time and event for one table*„" yang artinya anda tidak bisa membuat multiple trigger pada sebuah tabel pada waktu dan event yang sama.
- Solusi yang bisa dilakukan adalah menggabung isi trigger trig_update_mhs dengan isi trigger yang baru.

CONTOH 2

```
DELIMITER //
DROP TRIGGER IF EXISTS trig_update_mhs //
CREATE TRIGGER trig_update_mhs
AFTER UPDATE ON mhs
FOR EACH ROW
BEGIN
      IF OLD.alamat<>NEW.alamat THEN
        INSERT INTO history_alamat_mhs
           VALUES(now(),OLD.nim,OLD.alamat,USER());
      END IF;
      IF OLD.nim<>NEW.nim THEN
        UPDATE history_alamat_mhs
                SET nim=NEW.nim WHERE nim=OLD.nim;
      END IF;
END //
```

CONTOH 2

```
SELECT * FROM history_alamat_mhs
```

waktu	nim	alamat	oleh
2010-01-02 11:20:52	10107001	Jl. AA No. 1	root@localhost
2010-01-02 11:20:54	10107002	Jl. BB No. 2	root@localhost
2010-01-02 11:20:55	10107001	Jl. Dipati Ukur No. 1	root@localhost
2010-01-02 22:09:52	10107001	Jl. Merdeka No. 3	root@localhost

```
UPDATE mhs SET nim=10107010 WHERE nim=10107001
```

```
SELECT * FROM history_alamat_mhs
```

waktu	nim	alamat	oleh
2010-01-02 11:20:52	10107010	Jl. AA No. 1	root@localhost
2010-01-02 11:20:54	10107002	Jl. BB No. 2	root@localhost
2010-01-02 11:20:55	10107010	Jl. Dipati Ukur No. 1	root@localhost
2010-01-02 22:09:52	10107010	Jl. Merdeka No. 3	root@localhost

CONTOH 2

- Trigger pada contoh 2 bisa dimodifikasi untuk membuat trigger yang akan menghapus semua data pada tabel history_alamat_mhs ketika ada penghapusan pada tabel mhs.
- Hal ini bisa dilakukan dengan membuat trigger “AFTER DELETE ON mhs” yang akan menghapus semua data pada tabel history_alamat_mhs yang sesuai nim-nya dengan nim dari tabel mhs yang akan dihapus (“DELETE FROM history_alamat_mhs WHERE nim=OLD.nim”).

CONTOH 3

- Buatlah suatu trigger yang mencegah perubahan pada primary key tabel mhs (field nim). Jika ada perubahan, maka nim tidak boleh berubah. (agak kontradiksi dengan contoh 2, tapi nggak masalah. Sekedar contoh.).
- Hal ini dapat dilakukan yaitu dengan mengeset nilai nim yang baru (NEW.nim) dengan nilai yang nama (OLD.nim)

CONTOH 3

```
DELIMITER //
DROP TRIGGER IF EXISTS trig_update_nim_mhs //
CREATE TRIGGER trig_update_nim_mhs
BEFORE UPDATE ON mhs
FOR EACH ROW
BEGIN
     SET NEW.nim=OLD.nim;
END //
```

CONTOH 3

```
SELECT * FROM mhs
```

nim	nama	alamat
10107002	BEBE	Jl. Ir. H. Djuanda No. 100
10107003	CECE	Jl. CC No. 3
10107010	AAA	Jl. Suci No. 7

```
UPDATE mhs SET nim=10107111 WHERE nim=10107010
```

```
SELECT * FROM mhs
```

nim	nama	alamat
10107002	BEBE	Jl. Ir. H. Djuanda No. 100
10107003	CECE	Jl. CC No. 3
10107010	AAA	Jl. Suci No. 7

LATIHAN

- Buatlah sebuah tabel untuk menyimpan data transaksi transfer yang strukturnya sebagai berikut :
  - NoTransaksi : INT AUTO_INCREMENT
  - WaktuTransaksi : DATETIME
  - NoRekPengirim : INT, FK REF rekening(No)
  - NoRekPenerima : INT, FK REF rekening(No)
  - BesarTransfer : DOUBLE
- Jika ada penambahan data di tabel transfer (AFTER INSERT ON transfer), maka akan mengupdate saldo pada rekening yang bersangkutan sesuai dengan besarTransfer. Mirip dengan stored procedure Transfer.