

PRAKTIKUM SISTEM BASIS DATA

Modul 12: Trigger

Deskripsi

Modul ini berisi pengenalan tentang Trigger pada basis data MySQL. Trigger digunakan sebagai suatu fungsi yang dijalankan secara otomatis saat terjadi peristiwa insert, update, atau delete. Materi pada modul ini mencakup pengertian, keuntungan, kelemahan, dan implementasi trigger pada MySQL.

Tujuan dan Kompetensi Praktikum:

1. Mahasiswa dapat menjelaskan konsep Trigger pada Basis Data MySQL
2. Mahasiswa dapat membuat Trigger pada suatu tabel yang telah ditentukan
3. Mahasiswa dapat menjelaskan perbedaan macam-macam Trigger sesuai fungsinya

Alat dan Bahan:

1. Modul Praktikum
2. Komputer
3. MySQL dan phpmyadmin, atau DataGrip
Silakan membuat database terlebih dahulu, misalnya **[nama].p13**
4. Viewer LCD atau Zoom/Google Meet
5. Moodle

Materi dan Pembagian waktu:

Materi dan pembagian waktu pelaksanaan perkuliahan meliputi:

No.	Materi	Waktu
1	Pre-test	15 menit
2	Materi Trigger	45 menit
3	Latihan unguided	45 menit
4	Post-test	45 menit
	Total waktu	150 menit

Penilaian

Pada modul dilakukan pengambilan nilai pada pre dan post-test dan bonus pengerjaan latihan.

Materi Praktikum (45 menit)

1. Pengertian Trigger

Trigger termasuk dalam *stored* program yang khusus pada basis data. Trigger merupakan salah satu bentuk *stored procedure* (yang akan kita pelajari pada pertemuan berikutnya) tapi perilakunya sedikit berbeda. *Stored program* pada basis data adalah

suatu urutan perintah yang ditulis, disimpan, dan dieksekusi di dalam sebuah basis data server, dalam hal ini adalah MySQL server.

Trigger adalah *stored program* yang **diaktifkan (invoke)** sebagai akibat dari suatu aktivitas yang terjadi di dalam suatu basis data. Diaktifkan dalam bahasa Inggris disebut **di-trigger**. Biasanya trigger akan di-invoke akibat dari operasi DML (INSERT, UPDATE, DELETE) pada suatu tabel dalam basis data. Trigger dapat juga digunakan dalam validasi data. Perbedaan utama antara *stored procedure* dan trigger adalah bahwa trigger dipanggil/dieksekusi otomatis saat terjadi operasi DML pada suatu tabel, sedangkan *stored procedure* harus dipanggil secara eksplisit.

PERHATIAN:

Pada MySQL Trigger tidak akan dipanggil / diinvoke jika tidak ada perintah SQL yang diproses pada database server MySQL. Sehingga trigger tidak akan bisa dipanggil misalnya dengan menggunakan API yang tidak melalui MySQL.

Trigger diimplementasikan dalam bentuk **fungsi/prosedur** yang akan dijalankan saat terjadi perubahan pada tabel. Karena trigger berada di dalam suatu tabel, maka suatu program di luar basis data tidak akan bisa mem-bypass trigger. Artinya sekali trigger dibuat di dalam tabel, maka programmer yang membuat program dari luar basis data (misalnya kode PHP) tidak akan bisa melewati trigger yang sudah dibuat sebelumnya.

2. Keuntungan dan Kelemahan Trigger

Keuntungan Trigger:

- Trigger dapat menjadi cara alternatif untuk mengecek integritas data.
- Trigger dapat memvalidasi input data.
- Trigger dapat menangkap error pada aturan bisnis (*business logic*).
- Trigger dapat menjalankan program yang dijadwalkan karena secara otomatis dapat dipanggil saat terjadi operasi DML pada tabel.
- Trigger sangat berguna untuk mengaudit perubahan data pada tabel.

Kelemahan Trigger:

- Trigger hanya bisa menambah kemampuan validasi tambahan, tidak bisa menangani semua validasi. Artinya validasi utama tetap harus dibuat di level aplikasi (program), misalnya menggunakan PHP / Javascript, atau bahasa pemrograman lain.
- Trigger dipanggil dan dieksekusi secara tidak terlihat dari aplikasi client, sehingga tidak dapat diketahui secara pasti trigger apa yang dibuat di dalam tabel dari program.
- Trigger dapat menambah *overhead* pada basis data server, seperti MySQL.

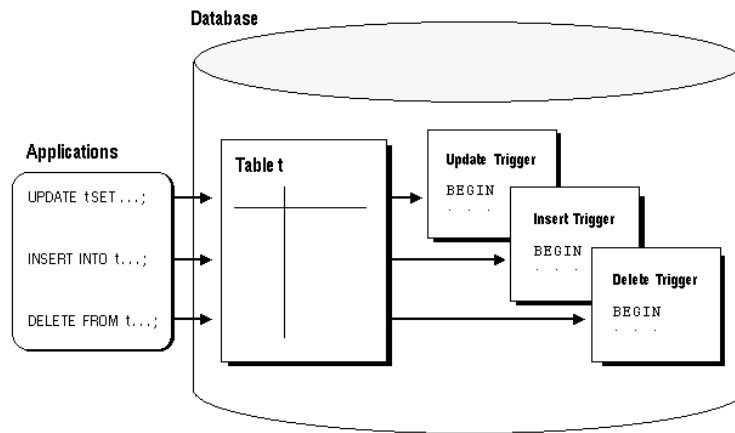
3. Implementasi Trigger

Pada MySQL terdapat 6 macam trigger:

- BEFORE INSERT – dipanggil sebelum data dimasukkan ke dalam tabel
- AFTER INSERT – dipanggil setelah data dimasukkan ke dalam tabel.
- BEFORE UPDATE – dipanggil sebelum data diupdate di dalam tabel.
- AFTER UPDATE – dipanggil setelah data diupdate di dalam tabel.
- BEFORE DELETE – dipanggil sebelum data dihapus di dalam tabel.
- AFTER DELETE – dipanggil setelah data dihapus di dalam tabel.

Pada Oracle, trigger bisa berisi SQL ataupun PL/SQL program.

Ilustrasi trigger adalah sebagai berikut:



Gambar 1. Trigger

Trigger biasanya digunakan untuk:

- Menghasilkan nilai kolom turunan (derived) secara otomatis
- Mencegah transaksi yang invalid / salah
- Memaksakan otorisasi yang kompleks
- Memaksakan referential integrity pada semua node dalam database terdistribusi
- Memaksakan aturan bisnis yang kompleks
- Menyediakan event logging yang transparan
- Menyediakan audit pada basis data
- Menjaga replikasi tabel agar tetap sinkron
- Mengumpulkan statistik terhadap akses terhadap suatu tabel

4. Pembuatan Trigger

Penamaan trigger mengikuti aturan berikut:

(BEFORE | AFTER)_tableName_(INSERT | UPDATE | DELETE)

Contoh nama trigger: before_mahasiswa_insert

Atau

tablename_(BEFORE | AFTER)_(INSERT | UPDATE | DELETE)

Contoh nama trigger versi dua: mahasiswa_after_update

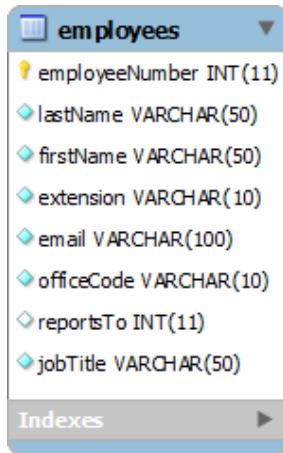
Sintaks pembuatan trigger pada MySQL:

```
CREATE TRIGGER trigger_name trigger_time trigger_event
ON table_name
FOR EACH ROW
BEGIN
...
END;
```

Keterangan:

- **Trigger name:** sesuai penamaan pada penjelasan sebelumnya
- **Trigger time:** before atau after
- **Trigger event:** event insert, atau update, atau delete

Perhatikan tabel berikut:



employees	
employeeNumber	INT(11)
lastName	VARCHAR(50)
firstName	VARCHAR(50)
extension	VARCHAR(10)
email	VARCHAR(100)
officeCode	VARCHAR(10)
reportsTo	INT(11)
jobTitle	VARCHAR(50)

Indexes

Berikut adalah SQL DDL create untuk tabel employeesaudit:

```
CREATE TABLE employeesaudit (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    employeeNumber INT NOT NULL,  
    firstName VARCHAR(50) NOT NULL,  
    lastName VARCHAR(50),  
    changeDate DATETIME DEFAULT NULL,  
    action VARCHAR(50) DEFAULT NULL  
);
```

Buat trigger pada tabel employeesaudit:

```
DELIMITER $$  
CREATE TRIGGER before_employee_update  
    BEFORE UPDATE ON employees  
    FOR EACH ROW  
    BEGIN  
        INSERT INTO employeesaudit  
        SET action = 'update',  
            employeeNumber = OLD.employeeNumber,  
            firstName = OLD.firstName,  
            lastName = OLD.lastName,  
            changeDate = NOW();  
    END$$  
DELIMITER ;
```

PERHATIAN:

Ada beberapa perbedaan sintaks dari INSERT/UPDATE/DELETE/SELECT ketika digunakan di dalam sebuah prosedur/trigger. Di sini terlihat bahwa ada SET dalam INSERT. Fungsi SET di sini serupa dengan VALUES, yang bedanya di SET diikuti dengan nama-nama atribut yang akan diisi nilainya.

Pada trigger update dan delete bisa digunakan keyword **OLD**, sedangkan pada trigger insert digunakan keyword **NEW** untuk mengakses data yang terkena trigger.

Trigger pada MySQL dapat dilihat dengan perintah:

SHOW TRIGGERS;

Coba tambahkan satu employees baru dengan data:

employeeNumber	1056
lastName	Filiana
firstName	Agatha
extension	123
Email	filiana@informatics.ukdw.ac.id
officeCode	100
reportsTo	1
jobTitle	MGR

Kemudian untuk menguji trigger, silakan coba melakukan update employee :

```
UPDATE employees
SET
    firstName = 'Agata'
WHERE
    employeeNumber = 1056;
```

Perubahan akan tersimpan di tabel employeesaudit.

Latihan UNGUIDED (45 menit)

1. Buatlah sebuah tabel bernama **buku**, yang berisi:

#	Name	Type	Collation	Attributes	Null	Default
1	kode_buku	varchar(5)	latin1_swedish_ci		No	None
2	nama_buku	varchar(100)	latin1_swedish_ci		No	None
3	pengarang	varchar(100)	latin1_swedish_ci		No	None
4	posisi_buku	varchar(4)	latin1_swedish_ci		No	None

Kemudian buat tabel **log_pindah_buku**, dengan ketentuan:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	log_id	int(11)			No	None		AUTO_INCREMENT
2	kode_buku	varchar(5)	latin1_swedish_ci		No	None		
3	posisi_lama	varchar(4)	latin1_swedish_ci		No	None		
4	posisi_baru	varchar(4)	latin1_swedish_ci		No	None		
5	waktu_pindah	datetime			No	None		

2. Buatlah trigger setelah insert data ke tabel buku:
Buat sebuah trigger pada tabel buku untuk mencatat kapan sebuah data buku ditambahkan ke dalam tabel log_pindah_buku. Masukkan kode_buku, posisi_lama = posisi awal yang dimasukkan, posisi_baru = posisi_lama, waktu_perubahan = waktu sekarang ke dalam log_pindah_buku menggunakan trigger.
3. Buatlah trigger setelah update data ke tabel buku:
Buat sebuah trigger pada tabel buku untuk mencatat perubahan posisi buku ketika sebuah record buku selesai diupdate ke dalam tabel log_pindah_buku. Masukkan kode_buku, posisi_lama = posisi_terakhir, posisi_baru = posisi_terbaru, waktu_pindah = waktu sekarang ke dalam log_pindah_buku menggunakan trigger.

4. Jalankan perintah insert ke tabel buku:
Jalankan 1 buah perintah untuk memasukkan sebuah item ke tabel buku. Buktikan dan capture bahwa trigger insert berjalan di tabel log_pindah_buku! Misalnya tambahkan produk dengan kode RE112 dengan judul "The Fault in Our Stars" pengarang "John Green" dan posisi buku di rak R-02.
5. Jalankan perintah update di tabel buku:
Jalankan 1 buah perintah untuk mengubah posisi rak buku suatu item buku ke tabel buku. Buktikan dan screenshot saat trigger update berjalan di tabel log_pindah_buku! Misalnya ubah posisi buku dengan kode RE112 ke rak buku R-33.
6. Buatlah trigger **before_buku_insert** untuk memeriksa validitas posisi rak buku sebelum diinsert, jika digit angka di posisi_buku = 0 harus diset ke 01. Tidak boleh juga lebih dari 50, jika lebih dari 50 maka harus diset ke 50. Huruf rak bersifat fleksibel (contoh: R-, K-, F-, dll)
(contoh 1: R-61 tidak valid sehingga harus diubah menjadi R-50)
(contoh 2: U-00 tidak valid sehingga harus diubah menjadi U-01)
Hint: gunakan operator dan fungsi manipulasi string serta konversi pada modul ke 8
7. Insert 2 buah buku. Buku pertama memiliki kode buku UM228 dengan judul "Divergent", pengarang "Veronica Roth", posisi buku di "U-00". Buku kedua memiliki kode buku RE112 dengan judul "Insurgent", pengarang "Veronica Roth", posisi buku di "M-77".

REFERENSI:

MySQL Trigger - <http://www.mysqltutorial.org/mysql-triggers.aspx>

Steven Feuerstein, Guy Harrison, "MySQL Stored Procedure Programming", 2006, O'Reilly. ISBN: 978-0-59-610089-6

Oracle Trigger:

https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm#intro%20triggers