BAB 11
TRIGGERING & STORAGE PROCEDURE

11.1. Tujuan Pembelajaran

1. Membuat triggering dan storage procedure
2. Mengetahui cara kerja triggering dan storage procedure
3. Mengetahui kegunaan triggering dan storage procedure

11.2. Dasar Teori

Pengenalan Trigger

Di MySQL, pemicu adalah program tersimpan yang dipanggil secara otomatis sebagai respons terhadap peristiwa seperti penyisipan, pembaruan, atau penghapusan yang terjadi di tabel terkait. Misalnya, Anda dapat menentukan pemicu yang dipanggil secara otomatis sebelum baris baru dimasukkan ke dalam tabel.

MySQL mendukung pemicu yang dipanggil sebagai respons terhadap kejadian INSERT, UPDATE, atau DELETE.

Standar SQL mendefinisikan dua jenis pemicu: row-level triggers(pemicu tingkat baris) dan statement-level triggers(pemicu tingkat pernyataan).

- *Row-level triggers*(pemicu tingkat baris) diaktifkan untuk setiap baris yang disisipkan, diperbarui, atau dihapus. Misalnya, jika tabel memiliki 100 baris yang disisipkan, diperbarui, atau dihapus, pemicu secara otomatis dipanggil 100 kali untuk 100 baris yang terpengaruh.
- *Statement-level triggers*(pemicu tingkat pernyataan) dijalankan sekali untuk setiap transaksi terlepas dari berapa banyak baris yang dimasukkan, diperbarui, atau dihapus.

MySQL hanya mendukung *Row-level triggers*(pemicu tingkat baris). Tetapi tidak mendukung *Statement-level triggers*(pemicu tingkat pernyataan).

Keuntungan Trigger

1. Pemicu menyediakan cara lain untuk memeriksa integritas data.
2. Pemicu menangani kesalahan dari lapisan basis data.
3. Pemicu memberikan cara alternatif untuk menjalankan tugas terjadwal. Dengan menggunakan pemicu, Anda tidak perlu menunggu acara terjadwal berjalan karena pemicu dipanggil secara otomatis sebelum atau setelah perubahan dilakukan pada data dalam tabel.
4. Pemicu dapat berguna untuk mengaudit perubahan data dalam tabel.

Kerugian Trigger

1. Pemicu hanya dapat memberikan validasi tambahan, tidak semua validasi. Untuk validasi sederhana, Anda dapat menggunakan batasan NOT NULL, UNIQUE, CHECK, dan FOREIGN KEY.
2. Pemicu bisa jadi sulit dipecahkan karena dijalankan secara otomatis di database, yang mungkin tidak terlihat oleh aplikasi klien.
3. Pemicu dapat meningkatkan overhead Server MySQL

Mengelola Trigger MySQL

- Create triggers – jelaskan langkah-langkah cara membuat pemicu di MySQL.
- Drop triggers – tunjukkan cara menjatuhkan pemicu.
- Create a BEFORE INSERT trigger – tunjukkan cara membuat pemicu BEFORE INSERT untuk mempertahankan tabel ringkasan dari tabel lain.
- Create an AFTER INSERT trigger – jelaskan cara membuat pemicu SETELAH INSERT untuk menyisipkan data ke dalam tabel setelah memasukkan data ke dalam tabel lain.
- Create a BEFORE UPDATE trigger – pelajari cara membuat pemicu BEFORE UPDATE yang memvalidasi data sebelum diperbarui ke tabel.
- Create an AFTER UPDATE trigger – menunjukkan kepada Anda cara membuat pemicu SETELAH PEMBARUAN untuk mencatat perubahan data dalam tabel.
- Create a BEFORE DELETE trigger – tunjukkan cara membuat pemicu BEFORE DELETE.
- Create an AFTER DELETE trigger – jelaskan cara membuat pemicu SETELAH DELETE.
- Show triggers – daftar pemicu dalam database, tabel berdasarkan pola tertentu.

Pengenalan Stored Procedure

Stored Procedure adalah sebuah fungsi berisi kode SQL yang dapat digunakan kembali. Dalam Stored Procedure juga dapat dimasukkan parameter sehingga fungsi dapat digunakan lebih dinamis berdasarkan parameter tersebut.

Stored procedure merupakan kumpulan dari perintah-perintah SQL yang tersimpan dalam suatu database, pada umumnya digunakan pada perintah DML(*select, insert, update*, dan *delete*). Pada sebuah *stored procedure* dapat diberikan parameter sesuai dengan kebutuhan, sehingga dengan menerapkan *stored procedure* yang disimpan didalam suatu database program aplikasi akan dapat digunakan lebih dinamis berdasarkan parameter yang diberikan, juga *stored procedure* dapat dikombinasikan dengan fungsi-fungsi pernyataan bersyarat dan berkondisi seperti fungsi IF, While.

Apa tujuan dari *stored procedure*? Salah satu tujuan atau kegunaan dari *stored procedure*, yaitu untuk mengefisienkan pembuatan code program pada suatu bahasa pemrograman ketika akan

membuat program CRUD(*Create, Read, Update, Delete*) menggunakan database MySQL. *Stored procedure* dibuat dan disimpan hanya satu kali pada sebuah database, selanjutnya dapat dipanggil kembali, sehingga prosedur dengan nama yang sama dalam program dapat lebih cepat dieksekusi. Jenis perintah-perintah dalam *stored procedure* meliputi menambah data(*insert*),mengubah data (*update*), menghapus data(*delete*), dan memilih/menyeleksi data (*select*).

Keuntungan Stored Procedure

1. Mengurangi lalu lintas jaringan, stored procedure membantu mengurangi lalu lintas jaringan antara aplikasi dan Server MySQL. Karena alih-alih mengirimkan banyak pernyataan SQL yang panjang, aplikasi hanya harus mengirim nama dan parameter prosedur tersimpan.
2. Pusatkan logika bisnis dalam database, kalian dapat menggunakan stored procedure untuk mengimplementasikan logika bisnis yang dapat digunakan kembali oleh beberapa aplikasi. stored procedure membantu mengurangi upaya duplikasi logika yang sama di banyak aplikasi dan membuat database Anda lebih konsisten.
3. Jadikan basis data lebih aman, administrator basis data dapat memberikan hak istimewa yang sesuai untuk aplikasi yang hanya mengakses stored procedure tertentu tanpa memberikan hak istimewa apa pun pada tabel yang mendasarinya.

Kerugian Stored Procedure

1. Penggunaan sumberdaya, Jika kalian menggunakan banyak prosedur tersimpan, penggunaan memori setiap sambungan akan meningkat secara substansial. Selain itu, penggunaan berlebihan sejumlah besar operasi logis dalam prosedur tersimpan akan meningkatkan penggunaan CPU karena MySQL tidak dirancang dengan baik untuk operasi logis.
2. Penyelesaian masalah, sulit untuk men-debug prosedur tersimpan. Sayangnya, MySQL tidak menyediakan fasilitas apa pun untuk men-debug stored procedure seperti produk database perusahaan lainnya seperti Oracle dan SQL Server.
3. Pemeliharaan, mengembangkan dan memelihara prosedur tersimpan seringkali membutuhkan keahlian khusus yang tidak dimiliki oleh semua pengembang aplikasi. Ini dapat menyebabkan masalah dalam pengembangan dan pemeliharaan aplikasi.

Sintak Dasar Stored Procedure

```
DELIMITER $$
CREATE PROCEDURE nama_procedure()
BEGIN
   kode      sql
END$$
DELIMITER ;
```

Pemanggilan Stored Procedure

CALL nama_procedure();

11.3. Software

- XAMPP
- MySQL

11.4. Pembelajaran trigger & storage procedure

11.4.1 Pembelajaran membuat update before trigger

Sintak Dasar Trigger

Pernyataan CREATE TRIGGER menciptakan pemicu baru. Berikut adalah sintaks dasar daripernyataan CREATE TRIGGER:

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
trigger_body;
```

Gambar 11.1 Sintak Dasar Membuat Trigger

Dalam sintaks ini:

- Pertama, tentukan nama pemicu yang ingin Anda buat setelah kata kunci CREATE TRIGGER. Perhatikan bahwa nama pemicu harus unik di dalam database.
- Selanjutnya, tentukan waktu tindakan pemicu yang dapat BEFORE atau AFTER yangmenunjukkan bahwa pemicu dipanggil sebelum atau setelah setiap baris diubah.
- Kemudian, tentukan operasi yang mengaktifkan pemicu, bisa berupa INSERT, UPDATE,atau DELETE.
- Setelah itu, tentukan nama tabel tempat pemicu berada setelah kata kunci ON.
- Terakhir, tentukan pernyataan yang akan dijalankan saat pemicu diaktifkan. Jika Anda ingin mengeksekusi banyak pernyataan, Anda menggunakan pernyataan majemuk BEGIN END.

Membuat Update Before Trigger

Pertama, buat tabel baru bernama orderdetails_audit untuk menyimpan perubahan pada tabel orderdetails:

```
CREATE TABLE orderdetails_audit (
    id INT AUTO_INCREMENT PRIMARY KEY,
    orderNumber INT NOT NULL,
    product_code VARCHAR(50) NOT NULL,
    action VARCHAR(50) DEFAULT NULL
);
```

Gambar 11.2 Query Membuat Table OrderDetails_Audit

Selanjutnya, buat pemicu BEFORE UPDATE yang dipanggil sebelum perubahan dibuat padatabel orderdetails

```
CREATE TRIGGER before_orderdetails_update
    BEFORE UPDATE ON orderdetails
    FOR EACH ROW
 INSERT INTO orderdetails_audit
 SET action = 'update',
     orderNumber = OLD.orderNumber,
     productCode = OLD.productCode;
```

Gambar 11.3 Query Membuat Trigger Before_OrderDetails_Update

Di dalam tubuh pemicu, gunakan kata kunci OLD untuk mengakses nilai kolom orderNumber dan productCode baris yang dipengaruhi oleh pemicu.

Kemudian, tampilkan semua pemicu di database saat ini dengan menggunakan pernyataanSHOW TRIGGERS:

Kueri SQL Anda berhasil dieksekusi.

```
SHOW TRIGGERS;
```

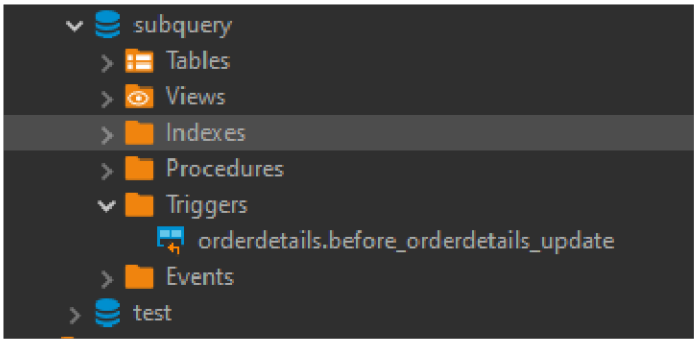
Gambar 11.4 Sintak Query Melihat Trigger

Outputnya akan menampilkan:

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer
before_orderdetails_update	UPDATE	orderdetails	INSERT INTO orderdetails_audit SET action = 'upd...	BEFORE	2022-12-08 06:19:08.12	NO_ZERO_IN_DATE,NO_ZERO_DATE,NO_ENGINE_SUBSTITUTIO...	root@lc

Gambar 11.5 Daftar Trigger Yang Sudah Dibuat

Selain itu, jika kalian ingin melihat skema menggunakan DBeaver. Klik subquery > triggers, Kalian akan melihat pemicu before_orderdetails_update seperti yang ditunjukkan pada gambar di bawah ini:



Gambar 11.6 Skema DBeaver

Setelah itu, perbarui baris di tabel orderdetails:

1 baris terpengaruh. (Pencarian dilakukan dalam 0,0006 detik.)

```
1 UPDATE orderdetails
2 SET orderdetails.productCode = "cnm08"
3 WHERE orderdetails.orderNumber = 11111;
```

Gambar 11.7 Query Update Table

Terakhir, kueri tabel orderdetails_audit untuk memeriksa apakah pemicu dipicu oleh pernyataan UPDATE:

Menampilkan baris 0 - 0 (total 1, Pencarian dilakukan dalam 0,0004 detik.)

```
1 SELECT * from orderdetails_audit;
```

Gambar 11.8 Query Menampilkan Tabel OrderDetails_Audit

Outputnya akan menampilkan:

id	orderNumber	productCode	action
1	11111	ad231	update

Gambar 11.9 Tampilan Tabel OrderDetails_Audit

Seperti yang terlihat dengan jelas dari output, pemicu secara otomatis dipanggil dan disisipkan baris baru ke dalam tabel employee_audit.

Menghapus Update Before Trigger

Pernyataan DROP TRIGGER digunakan untuk menghapus pemicu dari database. Berikut adalah sintaks dasar dari pernyataan DROP TRIGGER:

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name;
```

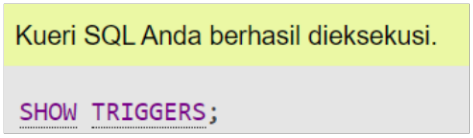
Gambar 11.10 Sintak Dasar Drop Trigger

Dalam sintaks ini:

1. Pertama, tentukan nama pemicu yang ingin kalian hapus setelah kata kunci DROP TRIGGER.
2. Kedua, tentukan nama skema atau database tempat pemicu itu berada. Jika kalian melewatkan nama skema, pernyataan tersebut akan menghapus pemicu di database saat ini.
3. Ketiga, gunakan opsi IF EXIST untuk melepaskan pemicu secara kondisional jika pemicunya ada. Klausa IF EXIST bersifat opsional.

Jika kalian melepaskan pemicu yang tidak ada tanpa menggunakan klausa IF EXIST, MySQL akan mengeluarkan *error*. Namun, jika kalian menggunakan klausa IF EXIST, MySQL akan mengeluarkan catatan sebagai gantinya.

DROP TRIGGER memerlukan hak istimewa TRIGGER untuk tabel yang terkait dengan pemicu. Cara menghapusnya, yaitu pertama kali yang harus dilakukan adalah melihat trigger sudah dibuat atau belum. Untuk mengeceknya kita bisa lakukan query seperti ini:



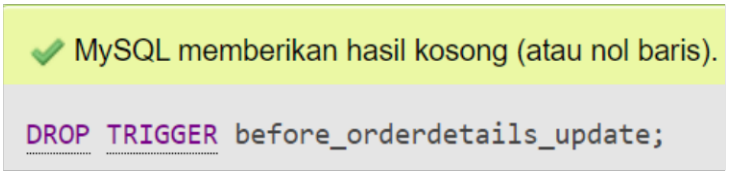
Gambar 11.11 Query Melihat Trigger

Jika sebelumnya trigger sudah dibuat, seharusnya akan memunculkan output seperti ini:

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character
before_orderdetails_update	UPDATE	orderdetails	INSERT INTO orderdetails_audit SET action = 'upd...	BEFORE	2022-12-09 04:02:14.94	NO_AUTO_VALUE_ON_ZERO	root@localhost	utf8mb4

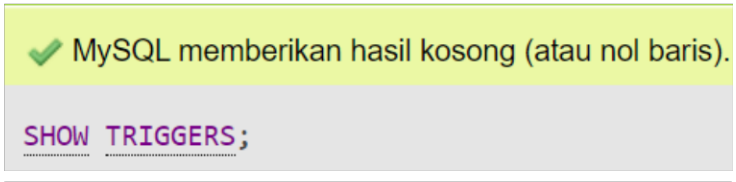
Gambar 11.12 List Trigger Yang Sudah Dibuat

Selanjutnya yang akan kita lakukan adalah menghapus trigger, kita bisa lakukan query sepertiini:



Gambar 11.13 Drop Trigger Before_OrderDetails_Update

Jika kita cek, apakah trigger sudah terhapus atau masih ada, lakukan query ini:



Gambar 11.14 Melihat Trigger

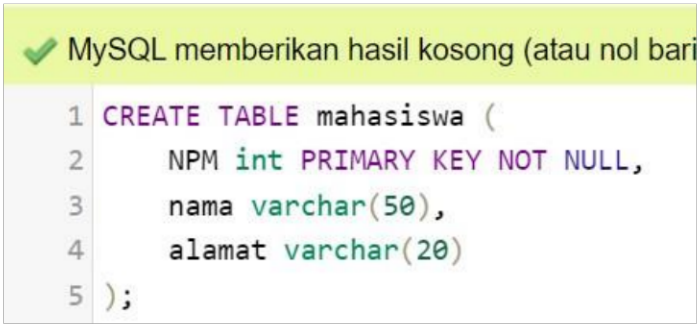
Jika query yang dibuat berhasil dijalankan dan tidak menampilkan apapun, berarti

triggeryang kita buat sudah berhasil dihapus,

11.4.2 Pembelajaran membuat storage procedure

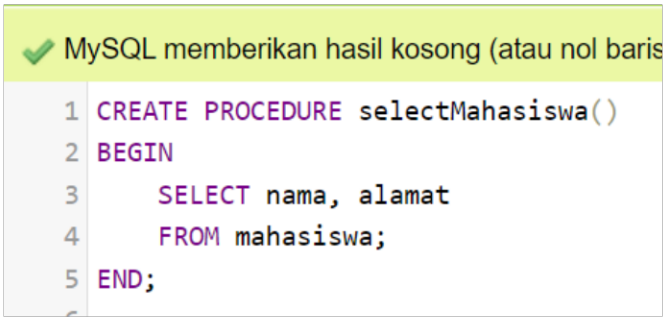
Membuat Membuat Storage Procedure

Pertama, buat dulu tabel yang akan digunakan untuk membuat storage procedure. Untuk contoh agar lebih mudah, disini digunakan tabel mahasiswa



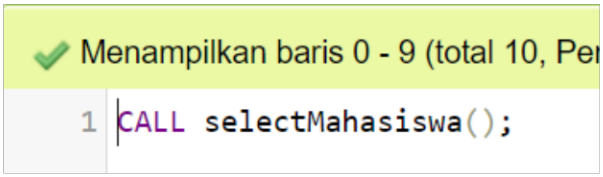
Gambar 11.15 Membuat Tabel Mahasiswa

Dan masukkan data di tabel mahasiswa dengan isian 10 data. Selanjutnya, kita akan membuat storage procedure untuk mendapatkan nama dan alamat mahasiswa, kita bisa melakukan query seperti ini:



Gambar 11.16 Membuat Procedure SelectMahasiswa

Untuk memanggil store procedure yang sudah dibuat, kita lakukan query:



Gambar 11.17 Memanggil Procedure SelectMahasiswa

Outputnya akan menampilkan:

NPM	nama	alamat
202310001	Anita	Bogor
202310003	Asep	Jakarta
202310005	Bani	Bandung
202310007	Caca	Tasikmalaya
202310009	Dani	Bogor
2023100011	Elisa	Jakarta
2023100013	Fahri	Bogor
2023100015	Gita	Bandung
2023100017	Hari	Bogor
2023100019	Ineu	Jakarta

Gambar 11.18 Tampilan Procedure SelectMahasiswa

Jadi, setiap ingin menampilkan nama dan alamat mahasiswa kita tidak perlu membuat kode SQL seperti biasanya. Cukup simpan kode SQL di Stored Procedure dan bisa kita panggildan gunakan berulang – ulang.

Membuat Storage Procedure Dengan Parameter

Kita juga memasukkan parameter di Stored Procedure agar menjadi lebih dinamis.Contoh,kita ingin membuat kode SQL untuk mencari data mahasiswa berdasarkan alamat.

```
CREATE PROCEDURE alamatMahasiswa
(
    alamatMhs VARCHAR(100)
)
BEGIN
    SELECT *
    FROM mahasiswa
    WHERE alamat = alamatMhs;

END;
```

Gambar 11.19 Membuat Procedure AlamatMahasiswa

Di dalam nama Stored Procedure terdapat parameter **alamatMhs** dengan tipe data varchar. Saat masuk ke kode SQL yaitu mencari mahasiswa dengan alamat yang sama dengan

parameter yang diinputkan

Cara memanggilnya kita bisa melakukan query seperti dibawah ini:

```
Menampilkan baris 0 - 1 (total 2, Pencarian d
1  CALL alamatMahasiswa("Bandung");
```

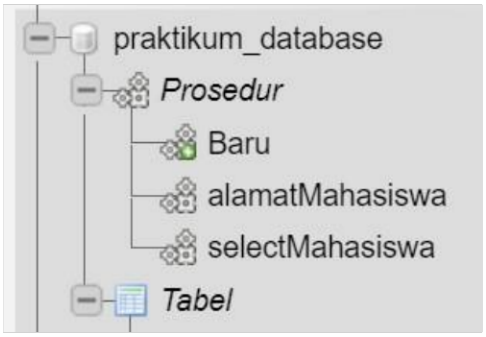
Gambar 11.20 Memanggil Procedure AlamatMahasiswa

Outputnya akan menampilkan:

NPM	nama	alamat
202310005	Bani	Bandung
2023100015	Gita	Bandung

Gambar 11.21 Tampilan Procedure AlamatMahasiswa

Kita bisa melihat storage procedure yang kita buat, dengan melihat menu disamping.



Gambar 11.22 Struktur Database

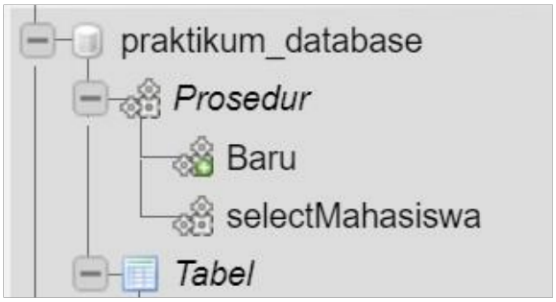
Jika kita ingin menghapus salah satu storage prosedur yang sudah dibuat, misal disini ingin menghapus storage procedure. Kita bisa melakukan query seperti ini:

```
MySQL memberikan hasil kosong (atau nol b

DROP PROCEDURE alamatMahasiswa;
```

Gambar 11.23 Menghapus Procedure AlamatMahasiswa

Dan jika kita cek lagi, maka storage procedure alamatMahasiswa sudah tidak tersedia atauterhapus.



Gambar 11.24 Procedure Setelah Dihapus

11.5. Latihan

1. Buatlah update after trigger!
2. Menurut kalian, praktis atau tidak membuat store procedure. Jelaskan!

Link latihan tambahan:
https://docs.google.com/document/d/1O4BNhvaOo-mTxmrPI4FV8KQmvnzNcd3obGtyt3zfmcQ/edit?usp=sharing