07

Stored Objects Pada MySQL

Bab ini membahas objek database yang disimpan yang didefinisikan dalam hal kode SQL yang disimpan di server untuk eksekusi nanti. Objek yang disimpan termasuk beberapa objek berikut :

- **Stored Procedure**
  Objek yang dibuat dengan CREATE PROCEDURE dan dipanggil menggunakan pernyataan CALL. Prosedur tidak memiliki nilai kembalian tetapi dapat mengubah parameternya untuk pemeriksaan selanjutnya oleh pemanggil.
- **Stored Function**
  Objek yang dibuat dengan CREATE FUNCTION dan digunakan seperti fungsi bawaan. Anda memanggilnya dalam ekspresi dan mengembalikan nilai selama evaluasi ekspresi.
- **Triger**
  Objek yang dibuat dengan CREATE TRIGGER yang dikaitkan dengan tabel. Pemicu diaktifkan saat peristiwa tertentu terjadi untuk tabel, seperti penyisipan atau pembaruan.
- **Event**
  Objek dibuat dengan CREATE EVENT dan dipanggil oleh server sesuai jadwal.
- **View**
  Objek yang dibuat dengan CREATE VIEW yang ketika direferensikan menghasilkan kumpulan hasil. Tampilan bertindak sebagai tabel virtual.

A. Stored Procedure

```
CREATE
    [DEFINER = user]
    PROCEDURE [IF NOT EXISTS] sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body
```

Keterangan :

```
proc_parameter:
    [ IN | OUT | INOUT ] param_name type

type:
    Any valid MySQL data type

characteristic: {
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
}

routine_body:
    Valid SQL routine statement
```

B. Store Function

```
CREATE
    [DEFINER = user]
    FUNCTION [IF NOT EXISTS] sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body
```

Keterangan :

```
func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic: {
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
}

routine_body:
    Valid SQL routine statement
```

C. Triger

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

atau

```
CREATE
    [DEFINER = user]
    TRIGGER [IF NOT EXISTS] trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    trigger_body
```

Keterangan :

trigger_time: { **BEFORE** | **AFTER** }

trigger_event: { **INSERT** | **UPDATE** | **DELETE** }

trigger_order: { **FOLLOWS** | **PRECEDES** } *other_trigger_name*

D. Event

```
CREATE
    [DEFINER = user]
    EVENT
    [IF NOT EXISTS]
    event_name
    ON SCHEDULE schedule
    [ON COMPLETION [NOT] PRESERVE]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'string']
    DO event_body;
```

Keterangan :

```
schedule: {
    AT timestamp [+ INTERVAL interval] ...
  | EVERY interval
    [STARTS timestamp [+ INTERVAL interval] ...]
    [ENDS timestamp [+ INTERVAL interval] ...]
}

interval:
    quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
              WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
              DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

E. View

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = user]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

F. Studi Kasus

1. Buat sebuah database dengan nama **siakad** table sesuai deskripsi di bawah ini :
   Buat 2 table yaitu table **mahasiswa** dan table **log_mahasiswa**
   - Buat Tabel mahasiswa *

NIM	Nama	Alamat
23402001	Andi	Madiun
23402002	Toni	Jakarta
23402003	Tina	Madiun

23402004	Tini	Ponorogo
23402005	Jaka	Magetan

**menyimpan data mahasiswa, nim(PK)*

- Table log_mahasiswa -> menyimpan perubahan data mahasiswa

Nama Kolom	Deskripsi
id_log	Int, Auto inc , PK
nim	Int(10)
alamat_baru	text
alamat_lama	text
waktu	date

Contoh Skenario **Trigger**:
Setiap ada perubahan data (UPDATE) alamat mahasiswa pada **table mahasiswa** akan disimpan/dicatat pada table sehingga log_mahasiwa berisi tentang histori perubahan data alamatyang terjadi.
Dengan adanya log perubahan data mahasiswa maka akan memudahkan dalam melihat histori data mahasiswa yang pernah berubah dalam sistem.

- Buat trigger :

```
DELIMITER $$
create Trigger alamat_mahasiswa_update
    before update  ON mahasiswa
    FOR each row
    Begin
        Insert into log_mahasiswa
        set nim= old.nim,
            alamat_lama= old.alamat,
            alamat_baru= new.alamat,
            waktu = now();
    END$$
DELIMITER ;
```

- Ubah data alamat pada tabel mahasiswa dan lihat hasil pada tabel log mahasiswa
- Tambahkan *trigger after delete* dan *after insert* untuk mencatat penghapusan data mahasiswa

```
DELIMITER $$
create Trigger alamat_mahasiswa_delete
    After Delete  ON mahasiswa
    FOR each row
    Begin
        Insert into log_mahasiswa
        set nim= old.nim,
            alamat_lama= old.alamat,
            waktu = now();
    END$$
DELIMITER ;
```

```
DELIMITER $$
create Trigger alamat_mahasiswa_insert
    after insert  ON mahasiswa
    FOR each row
    Begin
        Insert into log_mahasiswa
        set nim= new.nim,
            alamat_baru= new.alamat,
            waktu = now();
    END$$
DELIMITER ;
```

- Pelajari apa yang terjadi ketika ada penghapusan dan penambahan data

Contoh Skenario **Basic MySQL Stored procedures dan fungsi**:

- Membuat prosedure untuk menampilkan jumlah mahasiswa dengan nama *jumlah_mahasiswa()*

```
DELIMITER $$
CREATE PROCEDURE jumlah_mahasiswa (
    OUT jumlah INT
    )
    Begin
        SELECT COUNT(nim)   INTO jumlah
        from mahasiswa ;
    END$$
DELIMITER ;
```

Gunakan perintah berikut untuk melihat hasil:

```
CALL jumlah_mahasiswa(@total);
SELECT @total;
```

- Membuat fungsi untuk menampilkan jumlah mahasiswa

```
DELIMITER $$
CREATE function func_jumlah_mahasiswa ()
RETURNS int(10)
    Begin
        DECLARE jumlah INT;
        SET jumlah = 0;
        SELECT COUNT(nim) INTO jumlah from mahasiswa ;
        RETURN jumlah;
    END$$
DELIMITER ;
```

Tampilkan hasil

2. Buatlah impelementasi Triggger dan function/prosedure dalam DB study kasus masing2.
3. Buat aporan dan lampirkan screenshotnya