

# 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**
  - o 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\_mahasiswa berisi tentang histori perubahan data alamat yang terjadi.

Dengan adanya log perubahan data mahasiswa maka akan memudahkan dalam melihat histori data mahasiswa yang pernah berubah dalam sistem.

- Buat trigger :

```

1  DELIMITER $$
2  • create Trigger alamat_mahasiswa_update
3      before update ON mahasiswa
4      FOR each row
5      Begin
6          Insert into log_mahasiswa
7          set nim= old.nim,
8              alamat_lama= old.alamat,
9              alamat_baru= new.alamat,
10             waktu = now();
11      END$$
12  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 prosedur 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 implelementasi Triggger dan function/prosedure dalam DB study kasus masing2.
3. Buat aporan dan lampirkan screenshotnya

# 1. NOMOR 1 : Buat sebuah database dengan nama siakad

## a) Membuat Schema Database dan Trigger

```
1  -- MySQL Workbench Forward Engineering
2
3  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
4  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
5  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
6
7  -----
8  -- Schema all_siakad
9  -----
10
11  -----
12  -- Schema all_siakad
13  -----
14  CREATE SCHEMA IF NOT EXISTS `all_siakad` DEFAULT CHARACTER SET utf8 ;
15  USE `all_siakad` ;
16
17  -----
18  -- Table `all_siakad`.`mahasiswa`
19  -----
20  CREATE TABLE IF NOT EXISTS `all_siakad`.`mahasiswa` (
21    `NIM` INT(10) NOT NULL,
22    `Nama` VARCHAR(45) NULL,
23    `Alamat` TEXT NULL,
24    PRIMARY KEY (`NIM`))
25    ENGINE = InnoDB;
26
27
28  -----
29  -- Table `all_siakad`.`log_mahasiswa`
30  -----
31  CREATE TABLE IF NOT EXISTS `all_siakad`.`log_mahasiswa` (
32    `id_log` INT NOT NULL AUTO_INCREMENT,
33    `nim` INT(10) NULL,
34    `alamat_baru` TEXT NULL,
35    `alamat_lama` TEXT NULL,
36    `waktu` DATE NULL,
37    PRIMARY KEY (`id_log`))
38    ENGINE = InnoDB;
39
40  USE `all_siakad`;
41
42  DELIMITER $$
43  USE `all_siakad`$$
44  CREATE DEFINER = CURRENT_USER TRIGGER `all_siakad`.`mahasiswa_BEFORE_UPDATE` BEFORE UPDATE ON `mahasiswa` FOR EACH ROW
45  BEGIN
46    insert into log_mahasiswa
47      set nim = old.nim,
48          alamat_lama= old.alamat,
49          alamat_baru= new.alamat,
50          waktu = now();
51  END$$
52
53  USE `all_siakad`$$
54  CREATE DEFINER = CURRENT_USER TRIGGER `all_siakad`.`mahasiswa_AFTER_DELETE` AFTER DELETE ON `mahasiswa` FOR EACH ROW
55  BEGIN
56    insert into log_mahasiswa
57      set nim= old.nim,
58          alamat_lama= old.alamat,
59          waktu=now();
60  END$$
61
62  USE `all_siakad`$$
63  CREATE DEFINER = CURRENT_USER TRIGGER `all_siakad`.`mahasiswa_AFTER_INSERT` AFTER INSERT ON `mahasiswa` FOR EACH ROW
64  BEGIN
65    insert into log_mahasiswa
66      set nim= new.nim,
67          alamat_baru= new.alamat,
68          waktu=now();
69  END$$
70
71
72  DELIMITER ;
73
74  SET SQL_MODE=@OLD_SQL_MODE;
75  SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
76  SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

## b) Hasil

```
mysql> show tables;
+-----+
| Tables_in_all_siakad |
+-----+
| log_mahasiswa        |
| mahasiswa            |
+-----+
2 rows in set (0.01 sec)
```

```
mysql> insert into mahasiswa values  
-> (254311011,"Malik","Madiun");  
Query OK, 1 row affected (0.20 sec)
```

```
mysql> select * from mahasiswa;  
+-----+-----+-----+  
| NIM      | Nama  | Alamat |  
+-----+-----+-----+  
| 254311011 | Malik | Madiun |  
+-----+-----+-----+  
1 row in set (0.00 sec)
```

```
mysql> select * from log_mahasiswa;  
+-----+-----+-----+-----+-----+  
| id_log | nim      | alamat_baru | alamat_lama | waktu      |  
+-----+-----+-----+-----+-----+  
| 1      | 254311011 | Madiun      | NULL        | 2025-11-05 |  
+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

```
mysql> update mahasiswa set Alamat="Ngawi" where NIM=254311011;  
Query OK, 1 row affected (0.09 sec)  
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> select * from mahasiswa;  
+-----+-----+-----+  
| NIM      | Nama  | Alamat |  
+-----+-----+-----+  
| 254311011 | Malik | Ngawi  |  
+-----+-----+-----+  
1 row in set (0.00 sec)
```

```
mysql> select * from log_mahasiswa;  
+-----+-----+-----+-----+-----+  
| id_log | nim      | alamat_baru | alamat_lama | waktu      |  
+-----+-----+-----+-----+-----+  
| 1      | 254311011 | Madiun      | NULL        | 2025-11-05 |  
| 2      | 254311011 | Ngawi       | Madiun      | 2025-11-05 |  
+-----+-----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

```
mysql> delete from mahasiswa where NIM=254311011;  
Query OK, 1 row affected (0.05 sec)
```

```
mysql> select * from mahasiswa;  
Empty set (0.00 sec)
```

```
mysql> select * from log_mahasiswa;  
+-----+-----+-----+-----+-----+  
| id_log | nim      | alamat_baru | alamat_lama | waktu      |  
+-----+-----+-----+-----+-----+  
| 1      | 254311011 | Madiun      | NULL        | 2025-11-05 |  
| 2      | 254311011 | Ngawi       | Madiun      | 2025-11-05 |  
| 3      | 254311011 | NULL        | Ngawi       | 2025-11-05 |  
+-----+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

## 2. NOMOR 2 : Buatlah implemementasi Triggger dan function/prosedure dalam DB study kasus masing2

### a. Membuat Procedure dan Function

```
42  -----
43  -- procedure jumlah_mahasiswa
44  -----
45
46  DELIMITER $$
47  USE `all_siakad`$$
48  CREATE PROCEDURE `jumlah_mahasiswa` (out jumlah int)
49  BEGIN
50  select count(NIM) into jumlah from mahasiswa;
51  END$$
52
53  DELIMITER ;
```

```
42  -----
43  -- procedure jumlah_mahasiswa
44  -----
45
46  DELIMITER $$
47  USE `all_siakad`$$
48  CREATE PROCEDURE `jumlah_mahasiswa` (out jumlah int)
49  BEGIN
50  select count(NIM) into jumlah from mahasiswa;
51  END$$
52
53  DELIMITER ;
```

### b. Hasil

```
mysql> call jumlah_mahasiswa(@total);
Query OK, 1 row affected (0.09 sec)

mysql> select @total;
+-----+
| @total |
+-----+
|      0 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select func_jumlah_mahasiswa() ;
+-----+
| func_jumlah_mahasiswa() |
+-----+
|                        0 |
+-----+
1 row in set (0.02 sec)
```

