MySQL

TRIGGERS

Trigger di MySQL

BY SATRIA ARDI PERDANA

OVERVIEW

- Pengertian Trigger
- Kategori Trigger
- Even
- Syntax Trigger
- Create Trigger
- Show Trigger
- Edit dan Drop Trigger
- Timeout Lock Table
- Before dan After Insert Trigger
- Before dan After Update Trigger
- Before dan After Delete Trigger
- Create Multiple Trigger
- Schedule Event
- Alter Schedule Event
- Tanya Jawab
- Challenge Stok Trigger

MySQL

TRIGGERS

PENGERTIAN TRIGGER

- Trigger adalah script atau blok program SQL dalam suatu database yang direlasikan dengan satu atau beberapa tabel dalam database, akan aktif jika terpicu oleh suatu proses tertentu dalam blok code tersebut.

MySQL
TRIGGERS

KEGUNAAN TRIGGER

- Validasi data:
- Misal ada perhitungan yang menghasil kan nilai minus 5 (-5), sedangkan nilai minimal yang diharapkan adalah 0. maka bisa menggunakan trigger event BEFORE INSERT.
- Untuk uudit log dengan cara mengeksekusi script SQL secara otomatis.

  Audit log → proses mencatat suatu perubahan dalam database, misalnya :

  - Kolom dari tabel mana yang berubah
  - User / siapa yang melakukan perubahan.
  - Tanggal / kapan dilakukan perubahan.
- Untuk keamanan sistem
- misalnya untuk mencatat user siapa saja yang mengupdate ke tabel tertentu.
- Trigger dapat digunakan untuk mencatat history data dari suatu tabel yang critical. **tabel critical:** tabel yang sangat penting, misalnya tabel yang berisi data keuangan, data transaksi, data stok barang, data password dan lainnya.

MySQL
TRIGGERS

KATEGORI TRIGGER

Sesuai dengan SQL standar, trigger ada 2 kategori:

1. **Row level trigger** → trigger hanya akan berjalan jika setiap row terjadi proses insert, update, atau delete dalam suatu database. Jadi kalau terjadi proses even pada 100 row/baris pada suatu tabel maka akan ketrigger 100 x.

2. **Statement level trigger** → Trigger yang berjalan pada level **transaction**. Sekali jalan bisa memproses berapa/banyak (insert, uodate, delete).

Note: Pada MySQL hanya support row level trigger.

MySQL
TRIGGERS

KATEGORI TRIGGER

Row Level Trigger	Statement Level Trigger
Dieksekusi sekali untuk masing-masing dan setiap row dalam transaction.	Statement level triggers dieksekusi hanya sekali untuk setiap single transaction
Di syntax menggunakan **“FOR EACH ROW”** pada waktu **CREATE TRIGGER**	Di syntax menggunakan **“FOR EACH STATEMENT”** pada waktu **CREATE TRIGGER**
Contoh: Jika ada 100 row di insert ke dalam tabel, maka **row level trigger** akan dieksekusi sebanyak 100 kali.	Contoh: Jika ada 100 row di insert ke dalam tabel, maka **statement level trigger** hanya akan dieksekusi 1 kali.

MySQL
TRIGGERS

EVEN ROW LEVEL TRIGGER

- Event trigger: suatu kejadian yang akan dilakukan suatu trigger.
- Contoh : pada kasus ubah password user. Ketika user memproses update password, password lama akan disimpan dalam tabel backup (misalnya tabel password history) dan tabel yang sekarang akan berisi password yang baru.
- Event trigger:

EVENT	Keterangan
Before Insert	Trigger akan dijalankan sebelum data ditambahkan kedalam suatu tabel
After Insert	Trigger akan akan dijalankan setelah data ditambahkan kedalam suatu tabel
Before Update	Trigger akan dijalankan sebelum data dalam suatu tabel di update
After Update	Trigger akan dijalankan sesudah data dalam suatu tabel di update
Before Delete	Trigger akan dijalankan sebelum data dihapus dari suatu tabel
After Delete	Trigger akan dijalankan setelah data dihapus dari suatu tabel

MySQL
TRIGGERS

KEYWORD OLD DAN NEW DI TRIGGER

- OLD : mengacu pada nilai / record yang lama
- NEW : mengacu pada nilai / record yang baru

TRIGGER	OLD	NEW
INSERT	NO	YES
UPDATE	YES	YES
DELETE	YES	NO

SYNTAX TRIGGER

```
CREATE TRIGGER nama_trigger trigger_time trigger_event
   ON table_name
   FOR EACH ROW

BEGIN
     -- trigger body (sql statement)
END;
```

trigger_time	BEFORE | AFTER
trigger_event	INSER | UPDATE | DELETE

MySQL
TRIGGERS

BATASAN PADA TRIGGER

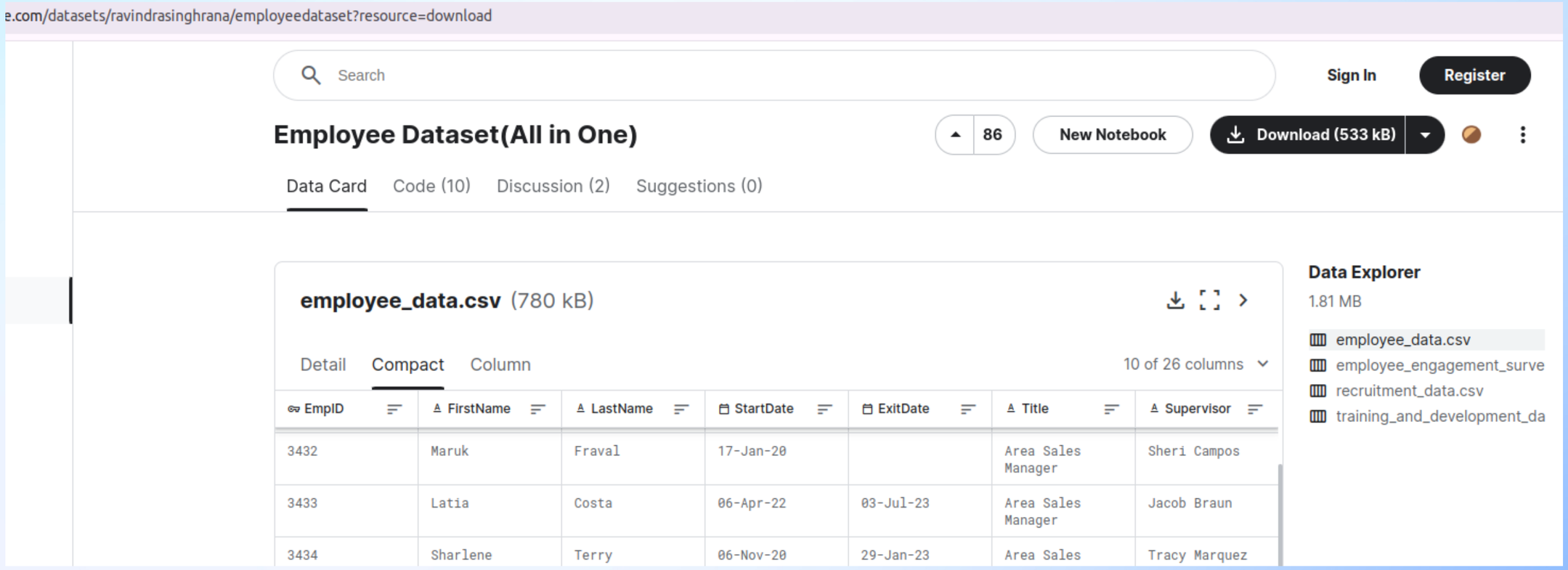
1. Satu trigger untuk setiap event → dalam trigger tidak bisa, misal event DELETE dan INSERT berbarengan, harus satu persatu dalam trogger.
2. Tidak bisa menggunakan **CALL statement** seperti store procedure ketika memanggilnya.
3. Trigger tidak bisa membuat temporary table atau view.

MySQL
TRIGGERS

SIAPKAN DATASET UNTUK MENCOBA TRIGGER

Digunakan dataset employee_data.csv yang di unduh dan di import kedalam MySQL seperti pada pertemuan sebelumnya(function).

https://www.kaggle.com/datasets/ravindrasinghrana/employeedataset?resource=download



MySQL
TRIGGERS

TABEL LOG UNTUK MENYIMPAN HASIL TRIGGER

Execute trigger tr_empl_log dibawah ini:

```
CREATE TABLE tb_master_employee_log (
  tb_log_id int NOT NULL AUTO_INCREMENT,
  tb_log_emp_id int DEFAULT NULL,
  tb_log_old_nama_depan varchar(100) DEFAULT NULL,
  tb_log_new_nama_depan varchar(100) DEFAULT NULL,
  tb_log_change_date datetime DEFAULT NULL,
  PRIMARY KEY (tb_log_id)
);
```

MySQL
TRIGGERS

CREATE TRIGGER

Buat trigger tr_empl_log dibawah ini:

```
CREATE TRIGGER tr_empl_log BEFORE UPDATE
     ON tb_master_employee
     FOR EACH ROW
BEGIN
     INSERT INTO tb_master_employee_log
     SET tb_log_emp_id = OLD.tbme_id,
          tb_log_old_nama_depan = OLD.tbme_firstname,
          tb_log_new_nama_depan = NEW.tbme_firstname,
          tb_log_change_date = now();
END;
```

Lakukan update pada tabel tb_master_employee untuk memastikan trigger bisa berjalan.

```
SELECT * FROM tb_master_employee; -- tabel sumber
SELECT * FROM tb_master_employee_log; -- tabel log jika trigger jalan

UPDATE tb_master_employee SET tbme_firstname = 'Uria' WHERE tbme_id = 3427; -- ex: update
```

MySQL
TRIGGERS

LANJUTAN CREATE TRIGGER

Hasil query tabel log setelah update statement dijalankan

```
SELECT * FROM tb_master_employee_log; -- tabel log jika trigger jalan
```

tb_master_employee_log 1

SELECT * FROM tb_master_empl | Enter a SQL expression to filter results (use Ctrl+Space)

	tb_log_id	tb_log_emp_id	tb_log_old_nama_depan	tb_log_new_nama_depan	tb_log_change_date
1	4	3,427	Uria	Uriah	2024-06-03 15:13:18
2	5	3,427	Uriah	Uria	2024-06-03 15:14:55

MySQL
TRIGGERS

SHOW TRIGGER

Script Show trigger:

```
SHOW TRIGGERS;
```

TRIGGERS 1

SHOW TRIGGERS | Enter a SQL expression to filter results (use Ctrl+Space)

	Trigger	Event	Table	Statement	Timing	
1	tr_empl_log	UPDATE	tb_master_employee	BEGIN ¶ INSERT INTO tb_master_employee_log¶ SE	AFTER	2

Script Show trigger menggunakan **IN**:

```
-- SHOW TRIGGERS IN nama_database;
   SHOW TRIGGERS IN pemrogramanbasisdata;
```

TRIGGERS 1

SHOW TRIGGERS IN pemrog | Enter a SQL expression to filter results (use Ctrl+Space)

	Trigger	Event	Table	Statement	Timing	
1	tr_empl_log_V2	UPDATE	pegawai	BEGIN ¶ INSERT INTO pegawai¶ SET kd_pegawai =	AFTER	2

MySQL
TRIGGERS

SHOW TRIGGER

Script Show trigger menggunakan **LIKE** yang daptat digunakan untuk mencari trigger yang tidak diinginkan:

```
-- SHOW TRIGGERS IN nama_database LIKE '%nama_tabel%';
   SHOW TRIGGERS IN view_procedure LIKE '%employee%'; -- bisa untuk cari issue trigger jalan yg tidak diharapkan.
```

ERS 1

V TRIGGERS IN view_procedure lik | Enter a SQL expression to filter results (use Ctrl+Space)

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_clie
tr_empl_log	UPDATE	tb_master_employee	BEGIN ¶ INSERT INT	AFTER	6-03 15:14:46.21	STRICT_TRANS_TAB	root@localhost	utf8mb4
tr_empl_log_test	UPDATE	tb_master_employee	BEGIN ¶ INSERT INT	AFTER	6-04 06:34:05.37	STRICT_TRANS_TAB	root@localhost	utf8mb4

Panels

MySQL
TRIGGERS

SHOW TRIGGER

Script Show trigger mencari berdasarkan event **INSERT** | **UPDATE** | **DELETE**

```
SHOW TRIGGERS IN view_procedure WHERE `event` = 'UPDATE';
```

TRIGGERS 1

SHOW TRIGGERS IN view_procedure WHE | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client
1	tr_empl_log	UPDATE	tb_master_em	BEGIN ¶ INSERT INT	AFTER	6-03 15:14:46.21	STRICT_TRANS_TAB	root@localhost	utf8mb4
2	tr_empl_log_test	UPDATE	tb_master_em	BEGIN ¶ INSERT INT	AFTER	6-04 06:34:05.37	STRICT_TRANS_TAB	root@localhost	utf8mb4

```
SHOW TRIGGERS IN view_procedure WHERE `event` = 'DELETE';
```

TRIGGERS 1

SHOW TRIGGERS IN view_procedure WHE | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client
1	tr_empl_log_dele	DELETE	tb_master_em	BEGIN ¶ INSERT INT	BEFORE	6-04 07:36:24.85	STRICT_TRANS_TAB	root@localhost	utf8mb4

MySQL
TRIGGERS

EDIT & DROP TRIGGER

CARA 1:

Untuk mengedit trigger dapat menggunakan perintah CREATE OR REPLACE TRIGGER, jadi setiap ada script baru dengan nama trigger yang sama akan mereplace script lama.

```
CREATE OR REPLACE TRIGGER tr_empl_log BEFORE UPDATE
     ON tb_master_employee
     FOR EACH ROW
BEGIN
     INSERT INTO tb_master_employee_log
     SET tb_log_emp_id = OLD.tbme_id,
          tb_log_old_nama_depan = OLD.tbme_firstname,
          tb_log_new_nama_depan = NEW.tbme_firstname,
          tb_log_change_date = now();
END;
```

MySQL
TRIGGERS

EDIT & DROP TRIGGER

Cara 2:

Drop dulu trigger lama baru create ulang trigger baru **dengan nama yang sama**.
Kenapa dengan nama yang sama ?

Bisa lihat demo delete trigger kemudian create ulang.

Ada tips ketika melakukan DELETE trigger maupun UPDATE trigger sebaiknya menggunakan cara dibawah ini:

1. Dilakukan di jam aman → sepi transaksi, tutup toko, after office hours.
2. Pastikan tidak ada aplikasi maupun client yang mengakses ke proses trigger.
3. Gunakan mekanisme lock table → supaya ngga terjadi masalah ketika create dan delete trigger. Kemudian setelah selesai jangan lupa di unlock.

MySQL
TRIGGERS

EDIT & DROP TRIGGER

Mekanisme lock TABLE ada 2, WRITE dan READ.

- READ: dengan READ user lain atau session lain yang mengakses ke suatu table yang di lock hanya bisa melakukan SELECT STATEMENT. Harus menunggu di UNLOCK.
- WRITE:dengan WRITE user lain atau session lain yang mengakses ke suatu table yang di lock tidak bisa melakukan apapun. Harus menunggu di UNLOCK.
- WRITE lebih tinggi tingkatannya dari pada READ, jadi jika ada 2 locking bersamaan maka READ akan menunggu WRITE terlebih dahulu.
- Bisa melihat demo contoh ketika lock dengan READ atau WRITE dengan syntax:

```
LOCK TABLES nama_tabel WRITE | READ;
LOCK TABLES tb_master_employee WRITE;
LOCK TABLES tb_master_employee READ;
UNLOCK TABLES;
```

MySQL
TRIGGERS

EDIT & DROP TRIGGER

Contoh menggunakan mekanisme lock ketika hapus trigger:

```
LOCK TABLES tb_master_employee WRITE;

DROP TRIGGER tr_empl_log; -- hapus trigger

CREATE TRIGGER tr_empl_log BEFORE UPDATE
     ON tb_master_employee
     FOR EACH ROW
BEGIN
     INSERT INTO tb_master_employee_log
     SET tb_log_emp_id = OLD.tbme_id,
          tb_log_old_nama_depan = OLD.tbme_firstname,
          tb_log_new_nama_depan = NEW.tbme_firstname,
          tb_log_change_date = now();
END;

UNLOCK TABLES; -- RELEASE LOCK
```

MySQL
TRIGGERS

TIMEOUT LOCK TABLE

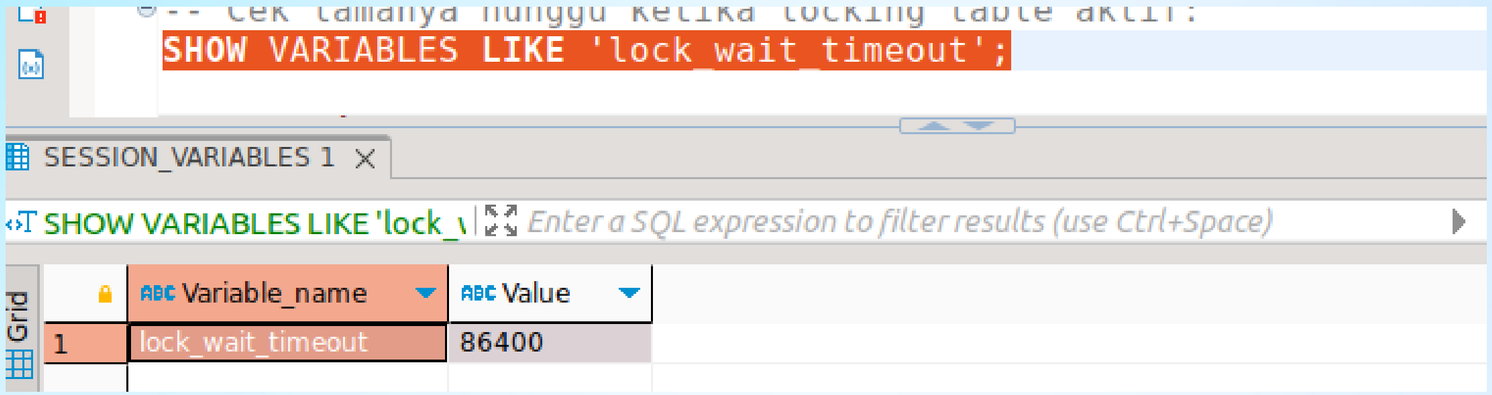
- Ketika melakukan mekanisme locking kepada suatu table, ada waktu timeoutnya sebelum mengirimkan pesan **error timeout.**

- Terjadi ketika suatu session login/user login melakukan, misal `LOCK TABLES tb_master_employee WRITE;,` maka user lain atau session lain tidak bisa melakukan mengakses (SELECT, INSERT, UPDATE atau DELETE) terhadap `tb_master_employee` selama waktu yang ditentukan pada `lock_wait_timeout`. Defaultnya selama 86400 second atau 24 jam.

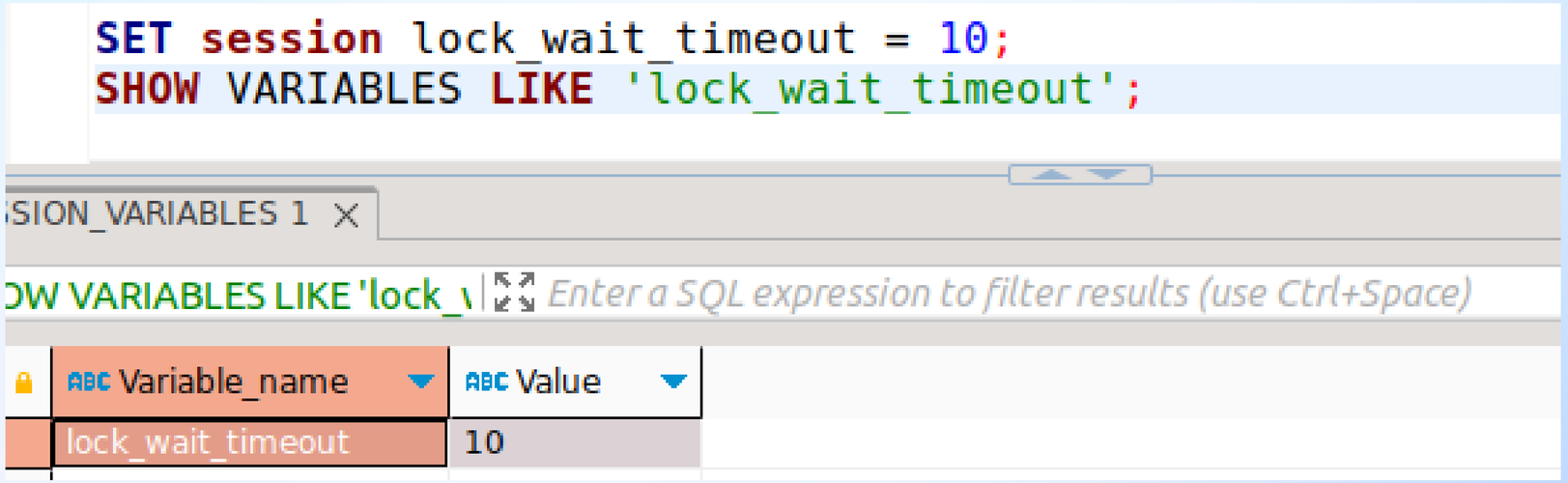
- Kita coba demo dengan merubah lama timeout `lock_wait_timeout = 10` second.

MySQL
TRIGGERS

DEMO TIMEOUT LOCK TABLE



Dirubah jadi seperti dibawah, kemudian lakukan lock terhadap tabel dan akses dengan user lain.



MySQL
TRIGGERS

BEFORE & AFTER INSERT TRIGGER

- Ketika menggunakan TRIGGER INSERT tidak bisa mengambil data **old**. Karena memang **INSERT** itu memasukkan data baru.
- Ketika menggunakan event **BEFORE INSERT** maka bisa melakukan validasi atau ubah data sebelum di **INSERT**.
- Untuk mempraktekan **BEFORE INSERT** perlu menambahkan kolom **tbme_health_limit** yang berisi jumlah dana yang bisa digunakan oleh employee untuk berobat.

```
-- Tambahkan field limit healt untuk mengetes AFTER INSERT
ALTER TABLE tb_master_employee ADD tbme_health_limit int;

UPDATE tb_master_employee SET tbme_health_limit = 10000000; -- isi limit ke 10 jt
```

MySQL
TRIGGERS

BEFORE INSERT TRIGGER

Buat trigger menggunakan event BEFORE INSERT

```
CREATE OR REPLACE TRIGGER tr_empl_before_insert BEFORE INSERT
     ON tb_master_employee
     FOR EACH ROW
BEGIN
     CASE WHEN NEW.tbme_current_employee_rating < 3 THEN
          SET NEW.tbme_current_employee_rating = 3;
     END CASE;
END;
```

MySQL
TRIGGERS

AFTER INSERT TRIGGER

- Buat tabel tb_master_employee_health_redeem yang digunakan untuk menyimpan history redeem limit kesehatan employee

```
CREATE TABLE tb_master_employee_health_redeem (
  tbme_id int(11) NOT NULL,
  tbme_amount_redeem int(11) DEFAULT NULL,
  tbme_created_date timestamp NULL DEFAULT current_timestamp()
);
```

- Buat trigger AFTER INSERT:

```
CREATE TRIGGER tr_empl_after_insert AFTER INSERT
     ON tb_master_employee_health_redeem
     FOR EACH ROW
BEGIN
     UPDATE tb_master_employee
          SET tbme_health_limit=(tbme_health_limit-NEW.tbme_amount_redeem)
     WHERE tbme_id=NEW.tbme_id;
END;
```

MySQL
TRIGGERS

LANJUTAN AFTER INSERT TRIGGER

- INSERT data ke tabel tb_master_employee_health_redeem untuk mengisi history amount kesehatan yang pernah dilakukan:

```
INSERT
    INTO tb_master_employee_health_redeem(tbme_id,tbme_amount_redeem,tbme_created_date)
    VALUES(3000,10000000,now());
```

- Cek Data menggunakan SELECT statement:

```
SELECT * FROM tb_master_employee_health_redeem;
SELECT * FROM tb_master_employee WHERE tbme_id =3000;
```

MySQL
TRIGGERS

BEFORE & AFTER UPDATE TRIGGER

Karakteristik EVENT UPDATE:

- Bisa mengambil OLD dan NEW sesuai dengan tabel di slide halaman 8.
- BEFORE UPDATE → bisa mengubah value data.
- AFTER UPDATE → tidak bisa mengubah value, karena ya memang data sudah masuk ke database.
- Pada contoh TRIGGER dibutuh kan **tb_master_employee** dan **tb_master_employee_log**

MySQL
TRIGGERS

BEFORE & AFTER UPDATE TRIGGER

BEFORE UPDATE

```
CREATE OR REPLACE TRIGGER tr_empl_before_update_log BEFORE UPDATE
     ON tb_master_employee
     FOR EACH ROW
BEGIN
     CASE WHEN NEW.tbme_current_employee_rating < 3 THEN
          SET NEW.tbme_current_employee_rating = 3;
     END CASE;
END;
```

AFTER UPDATE

```
CREATE OR REPLACE TRIGGER tr_empl_after_update_log AFTER UPDATE
     ON tb_master_employee
     FOR EACH ROW
BEGIN
     INSERT INTO tb_master_employee_log
     SET tb_log_emp_id = OLD.tbme_id,
          tb_log_old_nama_depan = OLD.tbme_firstname,
          tb_log_new_nama_depan = NEW.tbme_firstname,
          tb_log_change_date = now();
END;
```

MySQL
TRIGGERS

BEFORE & AFTER UPDATE TRIGGER

Jalankan UPDATE statement untuk mencoba BEFORE dan AFTER UPDATE sekaligus:

```
UPDATE tb_master_employee SET tbme_firstname = 'Susae',
tbme_current_employee_rating = 1 WHERE tbme_id = 1001; -- ex: update
SELECT DISTINCT * FROM tb_master_employee_log WHERE tb_log_emp_id =1001;
SELECT * FROM tb_master_employee;
```

MySQL
TRIGGERS

BEFORE & AFTER DELETE TRIGGER

BEFORE & AFTER DELETE :

- Bisa mengambil data OLD
- Tidak ada data NEW
- Tidak bisa mengedit value

PERBEDAAN BEFORE & AFTER DELETE TRIGGER::

- BEFORE DELETE TRIGGER bisa join ke tabel yang di delete.
- AFTER DELETE TRIGGER tidak bisa → karena memang data sudah dihapus.

MySQL
TRIGGERS

BEFORE & AFTER DELETE TRIGGER

Buat tabel **tb_master_employee_history** untuk mencoba BEFORE atau AFTER DELETE trigger.

```
CREATE TABLE tb_master_employee_history(
    tbme_id int primary key auto_increment,
     tbme_emp_id int,
     tbme_amount_redeem int,
     tbme_created_date timestamp
);
```

MySQL
TRIGGERS

BEFORE DELETE TRIGGER

BEFORE DELETE trigger:

```
CREATE OR REPLACE TRIGGER tr_empl_before_delete BEFORE DELETE
     ON tb_master_employee
     FOR EACH ROW
BEGIN
     INSERT INTO tb_master_employee_history
     SET tbme_emp_id = OLD.tbme_id,
          tbme_sisa_amount_redeem  = OLD.tbme_health_limit,
          tbme_created_date  = now();
END;
```

```
DELETE FROM tb_master_employee WHERE tbme_id =1003;
SELECT * FROM tb_master_employee;
SELECT * FROM tb_master_employee_history;
```

aster_employee_history 1

CT * FROM tb_master_employee_h | Enter a SQL expression to filter results (use Ctrl+Space)

tbme_id	tbme_emp_id	tbme_sisa_amount_redeem	tbme_created_date
8	1,003	10,000,000	2024-06-05 05:45:22

MySQL
TRIGGERS

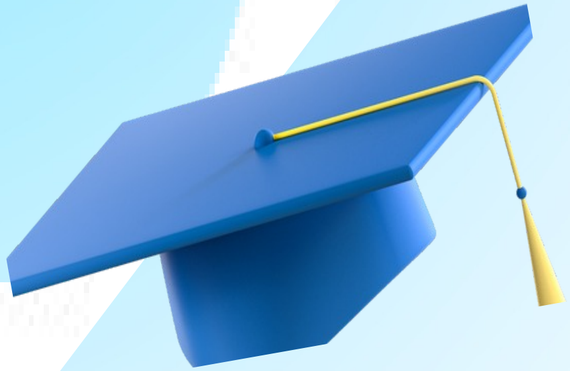
TRIGGER

Jika ada trigger yang bersamaan mengakses suatu tabel sebaiknya gunakan locking.

Jika ada beberapa trigger dengan nama yang berbeda tetapi didalamnya terdapat proses insert yang sama, maka semuanya akan jalan.

DO YOU HAVE ANY QUESTION?

Challenge
Soal / Task
Pengurangan Stock



THANK YOU

FOR YOUR ATTENTION