LAPORAN RESMI

MODUL VII

TRIGGER

SISTEM MANAJEMEN BASIS DATA

NAMA	**:**	**TESA PUTRI HUTAGAOL**
N.R.P	**:**	**220441100022**
DOSEN	**:**	**FITRI DAMAYANTI, S.Kom., M.Kom**
ASISTEN	**:**	**MUHAMMAD IQBAL FIRMANSYAH**
TGL PRAKTIKUM	**:**	**22 MEI 2024**

Disetujui : 04 Mei 2024
Asisten

MUHAMMAD IQBAL FIRMANSYAH
21.04.411.00084

LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Latar belakang dari pembuatan laporan ini adalah untuk pengenalan Trigger pada sistem manajemen basis data. Laporan praktikum ini disusun untuk memberikan penjelasan yang lebih terperinci mengenai penggunaan fitur yang ada dalam pengembangan basis data. Dalam sistem basis data tradisional, operasi bersifat pasif dan hanya merespons perintah eksplisit dari pengguna atau aplikasi melalui SQL, yang dapat membebani kontrol dan pemantauan data. Namun, dengan berkembangnya kebutuhan akan integritas data dan otomatisasi, penggunaan trigger dalam MySQL memungkinkan database untuk secara otomatis merespons operasi seperti penambahan, penghapusan, atau pembaruan data. Trigger memastikan integritas dan konsistensi data, mengotomatisasi proses bisnis, mencatat aktivitas pengguna untuk keperluan audit, dan memelihara data secara otomatis. Meskipun memiliki keterbatasan seperti tidak dapat dipanggil atau dibatalkan langsung dari program dan hanya mengizinkan satu trigger per kombinasi momen dan event per tabel, trigger memberikan fleksibilitas dan efisiensi yang lebih baik dalam pengelolaan data. Trigger juga dapat menggunakan variabel sistem seperti NEW dan OLD untuk akses data sebelum dan sesudah pembaruan, sehingga memperkuat pengendalian dan pemeliharaan data dalam database MySQL. Selain itu pembuatan laporan ini juga untuk memenuhi tugas mata kuliah Praktikum Sistem Manajemen Basis Data tentang *Trigger* Sistem Informasi Universitas Trunojoyo Madura.

1.2 Tujuan

- Mengenal trigger
- Mampu mendesain trigger sesuai dengan kebutuhan

BAB II
DASAR TEORI

2.1 Pendahuluan

Server database secara normal bersifat pasif. Database akan melakukan aksi ketika kita secara ekplisit memberikan perintah secara tertulis, misalnya melalui perintah SQL. Kita bisa menset agar database berubah dari passive menjadi aktif dengan menggunakan trigger.

Trigger adalah kode perintah SQL yang berisi perintah sql procedural dan perintah deklarative yang tersimpan di dalam database dan di aktifkan / dijalankan oleh server database jika sebuah operasi tertentu dijalankan didalam database. MySQL akan menjalankan trigger ketika ada program, atau user, atau store procedur yang menjalankan perintah database tertentu, yaitu ketika menambahkan baris data ke tabel atau ketika menghapus semua data dari tabel.

MySQL akan menjalankan trigger secara otomatis sesuai dengan kondisi tersebut. Trigger tidak bisa dipanggil atau di batalkan dari program.

Untuk mendefinisikan trigger menggunakan perintah CREATE TRIGGER, dengan diikuti dengan elemen trigger, yaitu :

1. trigger moment (before / after)
2. trigger event (insert, delete, update)
3. dan trigger action (yang dilakukan)

Contoh

```
CREATE TRIGGER INSERT_PLAYERS
   AFTER
   INSERT ON PLAYERS FOR EACH ROW
   BEGIN
      INSERT INTO CHANGES
         (USER, CHA_TIME, CHA_PLAYERNO,
          CHA_TYPE, CHA_PLAYERNO_NEW)
      VALUES (USER, CURDATE(), NEW.PLAYERNO, 'I', NULL);
   END
```

Trigger juga bisa memanggil store procedure

Trigger juga bisa memanggil stored procedure, misalnya

```
CREATE PROCEDURE INSERT_CHANGE
   (IN CPNO       INTEGER,
    IN CTYPE      CHAR(1),
    IN CPNO_NEW   INTEGER)
BEGIN
   INSERT INTO CHANGES (USER, CHA_TIME, CHA_PLAYERNO,
                        CHA_TYPE, CHA_PLAYERNO_NEW)
   VALUES (USER, CURDATE(), CPNO, CTYPE, CPNO_NEW);
END


CREATE TRIGGER INSERT_PLAYER
   AFTER INSERT ON PLAYERS FOR EACH ROW
   BEGIN
      CALL INSERT_CHANGE(NEW.PLAYERNO, 'I', NULL);
   END
```

trigger tidak bisa memiliki momen yang sama dan event yang sama dalam 1 tabel. Sebagai contoh, kita tidak bisa membuat 2 trigger BEFORE DELETE di tabel ‘mahasiswa’. Jika kita menginginkan ada 2 program untuk tabel tertentu, maka kita harus menggabungkannya dalam 1 trigger (bisa di pisah dituliskan dalam store procedure).

Ketika kita melakukam update record, ada 2 variabel yang muncul dalam sistem, NEW dan OLD. OLD menyimpan isi record dari data yang lama, dan NEW menyimpan isi record dari data yang baru. Kita bisa menggunakan 2 variabel ini di trigger.

Contoh

Contoh

```
CREATE TRIGGER DELETE_PLAYER
   AFTER DELETE ON PLAYERS FOR EACH ROW
   BEGIN
      CALL INSERT_CHANGE (OLD.PLAYERNO, 'D', NULL);
   END
```

Trigger juga bisa dimanfaatkan untuk melakukan pengecekan integrity constraint dan pengecekan data yang akan disimpan ke dalam tabel.

Contoh

```
CREATE TRIGGER BORN_VS_JOINED
   BEFORE INSERT, UPDATE ON PLAYERS FOR EACH ROW
   BEGIN
      IF YEAR(NEW.BIRTH_DATE) >= NEW.JOINED) THEN
         ROLLBACK WORK;
      END IF;
   END
```

05/06/2024

BAB III
TUGAS PENDAHULUAN

3.1 Soal

Dalam pengembangan perangkat lunak, pemahaman yang kuat tentang operasi database sangatlah penting. salah satu konsep utama adalah INSERT, UPDATE, dan DELEDE yang digunakan untuk mengelola data dalam database. selain itu, dalam konteks OLD dan NEW, yang terkait dengan operasi ini. Dalam esai ini, mari kita eksplor perbedaan antara INSERT, UPDATE, DELEDE, serta bagaimana konsep OLD dan NEW terkait dengan operasi-operasi tersebut

1. Bagaimana operasi DELETE berbeda dari INSERT dan UPDATE? Jelaskan dampak dari operasi DELETE pada data yang ada didalam tabel.
2. Bagaimana penggunaan konsep OLD dan NEW dalam memengaruhi logika bisnis pengembangan perangkat lunak? Berikan contoh skenario dimana pemahaman tentang OLD dan NEW diperlukan untuk mengimplementasikan logika bisnis yang kompleks.

3.2 Jawaban

1. DELETE menghapus data dari tabel secara permanen. sementara INSERT menambahkan data baru ke tabel dan UPDATE mengubah data yang sudah ada. konsep OLD dan NEW adalah nilai setelah diubah. operasi DELEDE tidak memunculkan konsep NEW karena data dihapus secara permanen, sehingga hanya ada nilai lama (OLD)
2. pemahaman OLD dan NEW penting buat sistem nangkep perubahan data. misalnya, diaplikasi manajemen stok barang, kalo harga barang diubah, sistem bisa dibandingkan hanya lama (OLD) sama yang baru (NEW). kalo hanya baru ngak bagus, sistem bisa balikin ke harga lama. ini membantu sistem mengambil keputusan berdasarkan perubahan data, jadi logika bisnis bisa berjalan lancar.

05/06/2021

Lifecycle whatsapp

1. Buka (on create) whatsapp dimulai dan siap digunakan
2. lanjut (onstart & onresume) whatsapp aktif dan bisa dipakai
3. pause (onpause) andi pindah ke mobile legends, whatsapp dipause.
4. Stop (onstop) whatsapp tidak terlihat dan di background
5. Henti paksa (on Destry) sistem butuh memori, whatsapp di hentikan. saat andi balik, harus memulai lagi dari awal.

BAB IV
IMPLEMENTASI

4.1 Soal

1. Buatlah sebuah trigger bernama after_update_transaksi yang akan memperbarui status mobil menjadi tersedia setiap kali status transaksi berubah menjadi selesai!
2. Buatlah sebuah trigger bernama after_insert_pembayaran yang akan mencatat log pembayaran ke dalam tabel log_pembayaran setiap kali ada pembayaran baru yang ditambahkan ke tabel pembayaran. Sebelum membuat trigger, buatlah terlebih dahulu tabel log_pembayaran yang akan digunakan untuk menyimpan log tersebut! Ketentuan tabel log_pembayaran:
    a. (id_log) primary key dengan tipe data INT dan auto increment.
    b. (id_pembayaran) dengan tipe data INT, tidak boleh NULL.
    c. (id_transaksi) dengan tipe data INT, tidak boleh NULL.
    d. (tanggal_pembayaran) dengan tipe data DATE, tidak boleh NULL.
    e. (jumlah_pembayaran) dengan tipe data DECIMAL(10, 2), tidak boleh NULL.
    f. (metode_pembayaran) dengan tipe data VARCHAR(50), tidak boleh NULL.
    g. (timestamp) dengan tipe data TIMESTAMP dan default value CURRENT TIMESTAMP.
3. Buatlah sebuah trigger bernama before_insert_transaksi yang akan menghitung total biaya sewa secara otomatis sebelum transaksi baru dimasukkan ke dalam tabel transaksi. Total biaya dihitung berdasarkan harga sewa per hari dari tabel mobil dikalikan dengan jumlah hari sewa (tanggal_selesai - tanggal_mulai).
4. Buatlah sebuah trigger bernama after_delete_transaksi yang akan mencatat log setiap kali ada transaksi yang dihapus dari tabel transaksi. Log ini harus mencatat id_transaksi, id_pelanggan, id_mobil, tanggal_mulai, tanggal_selesai, total_biaya, dan timestamp penghapusan. Ketentuan tabel log_hapus_transaksi:

a. (id_log) sebagai primary key dengan tipe data INT dan auto increment.
b. (id_transaksi) dengan tipe data INT, tidak boleh NULL.
c. (id_pelanggan) dengan tipe data INT, tidak boleh NULL.
d. (id_mobil) dengan tipe data INT, tidak boleh NULL.
e. (tanggal_mulai) dengan tipe data DATE, tidak boleh NULL.
f. (tanggal_selesai) dengan tipe data DATE, tidak boleh NULL.
g. (total_biaya) dengan tipe data DECIMAL(10, 2), tidak boleh NULL.
h. (timestamp) dengan tipe data TIMESTAMP dan default value CURRENT TIMESTAMP..

4.2 Query

1. Soal 1

```
 CREATE DATABASE rental_mobil;
USE rental_mobil;
DROP DATABASE rental_mobil;

CREATE TABLE pegawai(
        id_pegawai INT (11) PRIMARY KEY AUTO_INCREMENT NOT NULL,
        nama VARCHAR (50) NOT NULL,
        jabatan VARCHAR (50) NOT NULL,
        no_telepon VARCHAR (15)NOT NULL,
        email VARCHAR (50) NOT NULL
);
DESC pegawai;

INSERT INTO pegawai (nama, jabatan, no_telepon, email) VALUES
        ('Andi Wijaya', 'Manager', '081234567890', 'andi.wijaya@gmail.com'),
        ('Budi Santoso', 'Supervisor', '081298765432', 'budi.santoso@gmail.com'),
        ('Citra Dewi', 'Staff', '081212345678', 'citra.dewi@gmail.com'),
        ('Dedi Supriyadi', 'Clerk', '081276543210', 'dedi.supriyadi@gmail.com'),
```

```
        ('Eka Putri', 'Administrator', '081234098765',
'eka.putri@gmail.com'),
        ('Fajar Rahman', 'Manager', '081234567891',
'fajar.rahman@gmail.com'),
        ('Gita Permata', 'Supervisor', '081298765433',
'gita.permata@gmail.com'),
        ('Hari Santosa', 'Staff', '081212345679',
'hari.santosa@gmail.com'),
        ('Indra Wijaya', 'Clerk', '081276543211',
'indra.wijaya@gmail.com'),
        ('Joko Susilo', 'Administrator', '081234098766',
'joko.susilo@gmail.com');
SELECT * FROM pegawai;


CREATE TABLE pelanggan (
        id_pelanggan INT (11) PRIMARY KEY AUTO_INCREMENT
NOT NULL,
        nama VARCHAR (100) NOT NULL,
        alamat TEXT NOT NULL,
        no_telepon VARCHAR (15)NOT NULL,
        email VARCHAR (50) NOT NULL
);
DESC pelanggan;


INSERT INTO pelanggan (nama, alamat, no_telepon, email) VALUES
        ('Ahmad Fauzi', 'Jl. Merdeka No. 45, Jakarta', '081234567890',
'ahmad.fauzi@gmail.com'),
        ('Budi Hartono', 'Jl. Sudirman No. 10, Bandung', '081298765432',
'budi.hartono@gmail.com'),
        ('Citra Lestari', 'Jl. Diponegoro No. 20, Surabaya',
'081212345678', 'citra.lestari@gmail.com'),
        ('Dewi Sartika', 'Jl. Gatot Subroto No. 15, Medan',
'081276543210', 'dewi.sartika@gmail.com'),
        ('Eko Prasetyo', 'Jl. Thamrin No. 5, Yogyakarta', '081234098765',
'eko.prasetyo@gmail.com'),
        ('Fani Rahayu', 'Jl. Dipatiukur No. 30, Bandung', '081234567891',
'fani.rahayu@gmail.com'),
        ('Guntur Widodo', 'Jl. Ahmad Yani No. 25, Surabaya',
'081298765433', 'guntur.widodo@gmail.com'),
        ('Hani Susanti', 'Jl. Raden Saleh No. 35, Jakarta', '081212345679',
'hani.susanti@gmail.com'),
```

```
        ('Indra Kusuma', 'Jl. Pahlawan No. 40, Semarang',
'081276543211', 'indra.kusuma@gmail.com'),
        ('Joko Purnomo', 'Jl. A. Yani No. 50, Malang', '081234098766',
'joko.purnomo@gmail.com');
SELECT * FROM pelanggan;

CREATE TABLE mobil (
        id_mobil INT PRIMARY KEY AUTO_INCREMENT NOT
NULL,
        merk VARCHAR (50) NOT NULL,
        model VARCHAR (50) NOT NULL,
        tahun INT (11) NOT NULL,
        warna VARCHAR (20) NOT NULL,
        harga_sewa DECIMAL (10,2) NOT NULL,
        STATUS ENUM ('Dipinjam','Tersedia')
);
DESC mobil;

INSERT INTO mobil (merk, model, tahun, warna, harga_sewa,
STATUS) VALUES
        ('Toyota', 'Avanza', 2021, 'Putih', 500000.00, 'Dipinjam'),
        ('Honda', 'Civic', 2020, 'Hitam', 750000.00, 'Dipinjam'),
        ('Suzuki', 'Ertiga', 2019, 'Merah', 600000.00, 'Tersedia'),
        ('Mitsubishi', 'Xpander', 2022, 'Abu-abu', 700000.00, 'Tersedia'),
        ('Daihatsu', 'Terios', 2021, 'Biru', 550000.00, 'Dipinjam'),
        ('Nissan', 'Grand Livina', 2020, 'Silver', 650000.00, 'Tersedia'),
        ('Honda', 'HR-V', 2019, 'Putih', 800000.00, 'Dipinjam'),
        ('Toyota', 'Innova', 2022, 'Hitam', 900000.00, 'Tersedia'),
        ('Suzuki', 'XL7', 2021, 'Biru', 700000.00, 'Tersedia'),
        ('Mitsubishi', 'Pajero Sport', 2020, 'Coklat', 1200000.00,
'Dipinjam');
SELECT * FROM mobil;

CREATE TABLE perawatan (
        id_perawatan INT (11) PRIMARY KEY AUTO_INCREMENT
NOT NULL,
        id_mobil INT (11)NOT NULL ,
        tanggal DATE NOT NULL,
```

```
        deskripsi TEXT NOT NULL ,
        biaya DECIMAL (10,2) NOT NULL,
        FOREIGN KEY (id_mobil)REFERENCES mobil(id_mobil)
);
DESC perawatan;

INSERT INTO perawatan (id_mobil, tanggal, deskripsi, biaya)
VALUES
        (1, '2023-01-15', 'Ganti oli dan pemeriksaan rutin', 300000.00),
        (2, '2023-02-20', 'Penggantian ban dan balancing', 1500000.00),
        (3, '2023-03-10', 'Perbaikan rem dan pengecekan mesin',
500000.00),
        (4, '2023-04-05', 'Servis berkala dan penggantian filter udara',
700000.00),
        (5, '2023-05-25', 'Penggantian aki dan pemeriksaan listrik',
1200000.00),
        (6, '2023-06-10', 'Ganti oli dan tune-up mesin', 400000.00),
        (7, '2023-06-20', 'Penggantian kampas rem dan servis rem',
800000.00),
        (8, '2023-07-05', 'Pemeriksaan rutin dan penggantian kampas
kopling', 600000.00),
        (9, '2023-07-15', 'Penggantian shockbreaker dan penyetelan
suspensi', 900000.00),
        (10, '2023-08-01', 'Pemeriksaan sistem pendingin dan
penggantian radiator', 750000.00);
SELECT * FROM perawatan;

CREATE TABLE transaksi (
        id_transaksi INT (11) PRIMARY KEY AUTO_INCREMENT
NOT NULL,
        id_pelanggan INT (11) NOT NULL,
        id_mobil INT (11) NOT NULL,
        id_pegawai INT (11) NOT NULL,
        tanggal_mulai DATE NOT NULL,
        tanggal_Selesai DATE NOT NULL,
        total_biaya DECIMAL (10,2) NOT NULL,
        status_transaksi ENUM ('Belum Selesai','Selesai') NOT NULL,
        FOREIGN KEY (id_pelanggan) REFERENCES
pelanggan(id_pelanggan),
```

```
        FOREIGN KEY (id_mobil) REFERENCES mobil(id_mobil),
        FOREIGN KEY (id_pegawai) REFERENCES
pegawai(id_pegawai)
);
DESC transaksi;



INSERT INTO transaksi (id_pelanggan, id_mobil, id_pegawai,
tanggal_mulai, tanggal_selesai, total_biaya, status_transaksi) VALUES
        (1, 1, 1, '2024-05-10', '2024-05-13', 1500000.00, 'Selesai'),
        (2, 2, 2, '2024-05-11', '2024-05-15', 3000000.00, 'Belum Selesai'),
        (3, 3, 3, '2024-05-12', '2024-05-12', 600000.00, 'Selesai'),
        (4, 4, 4, '2024-05-13', '2024-05-15', 1400000.00, 'Belum Selesai'),
        (5, 5, 5, '2024-05-14', '2023-05-16', 1100000.00, 'Belum Selesai'),
        (6, 6, 6, '2024-05-15', '2023-05-20', 3250000.00, 'Selesai'),
        (7, 7, 7, '2024-05-16', '2023-05-19', 2400000.00, 'Belum Selesai'),
        (8, 8, 8, '2024-05-17', '2023-05-21', 3600000.00, 'Selesai'),
        (9, 9, 9, '2024-05-19', '2023-05-23', 2800000.00, 'Belum Selesai'),
        (10, 10, 10, '2024-05-22', '2023-05-24', 2400000.00, 'Belum
Selesai');
SELECT * FROM transaksi;

CREATE TABLE pembayaran (
        id_pembayaran INT (11) PRIMARY KEY
AUTO_INCREMENT NOT NULL,
        id_transaksi INT(11) NOT NULL,
        tanggal_pembayaran DATE NOT NULL,
        jumlah_pembayaran DECIMAL (10,2),
        metode_pembayaran VARCHAR (50),
        FOREIGN KEY (id_transaksi) REFERENCES transaksi
(id_transaksi)
);
DESC pembayaran;

INSERT INTO pembayaran (id_transaksi, tanggal_pembayaran,
jumlah_pembayaran, metode_pembayaran) VALUES
   (1, '2024-05-13', 1500000.00, 'Tunai'),
   (2, '2024-05-15', 1000000.00, 'Kartu Kredit'),
```

```
   (3, '2024-05-12', 600000.00, 'Transfer Bank'),
   (4, '2024-05-14', 700000.00, 'Tunai'),
   (5, '2024-05-17', 800000.00, 'Kartu Debit'),
   (6, '2024-05-20', 3250000.00, 'Transfer Bank'),
   (7, '2024-05-18', 2000000.00, 'Tunai'),
   (8, '2024-05-21', 3600000.00, 'Kartu Kredit'),
   (9, '2024-05-24', 2500000.00, 'Transfer Bank');
   -- (10, '2024-05-25', 2400000.00, 'Kartu Debit');
SELECT * FROM pembayaran;

-- NOMOR1--
DELIMITER //
CREATE TRIGGER after_update_transaksi
AFTER UPDATE ON transaksi
FOR EACH ROW
BEGIN
   IF NEW.status_transaksi = 'Selesai' THEN
      UPDATE mobil
      SET STATUS = 'Tersedia'
      WHERE id_mobil = NEW.id_mobil;
   END IF;
END //
DELIMITER ;
UPDATE transaksi SET status_transaksi='Selesai' WHERE id_transaksi
= 12;
SELECT * FROM transaksi;
SELECT * FROM mobil;



-- NOMOR2--
CREATE TABLE log_pembayaran (
   id_log INT AUTO_INCREMENT PRIMARY KEY,
   id_pembayaran INT NOT NULL,
   id_transaksi INT NOT NULL,
   tanggal_pembayaran DATE NOT NULL,
   jumlah_pembayaran DECIMAL(10, 2) NOT NULL,
   metode_pembayaran VARCHAR(50) NOT NULL,
```

```
   TIMESTAMP TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
DESC log_pembayaran;
SELECT * FROM pembayaran;
SELECT * FROM transaksi;

DELIMITER //
CREATE TRIGGER after_insert_pembayaran
        AFTER INSERT
        ON pembayaran
        FOR EACH ROW
BEGIN
        -- select * from log_pembayaran
        INSERT INTO log_pembayaran
(id_pembayaran,id_transaksi,tanggal_pembayaran,jumlah_pembayaran,
metode_pembayaran)
        VALUES
(new.id_pembayaran,new.id_transaksi,new.tanggal_pembayaran,new.ju
mlah_pembayaran,new.metode_pembayaran);
END //
DELIMITER ;
INSERT INTO pembayaran (id_transaksi, tanggal_pembayaran,
jumlah_pembayaran, metode_pembayaran) VALUES
(10, '2024-05-24', 240000.00, 'Kartu Debit');
SELECT * FROM pembayaran;
SELECT * FROM log_pembayaran;



-- NOMOR3--
DELIMITER //

CREATE TRIGGER before_insert_transaksi
BEFORE INSERT ON transaksi
FOR EACH ROW
BEGIN
   DECLARE harga_per_hari DECIMAL(10, 2);
   DECLARE jumlah_hari INT;
```

```
    -- Mendapatkan harga sewa per hari dari tabel mobil
    SELECT harga_sewa INTO harga_per_hari
    FROM mobil
    WHERE id_mobil = NEW.id_mobil;

    -- Menghitung jumlah hari sewa, pastikan tanggal tidak null
    IF NEW.tanggal_mulai IS NOT NULL AND NEW.tanggal_selesai IS
NOT NULL THEN
        SET jumlah_hari = DATEDIFF(NEW.tanggal_selesai,
NEW.tanggal_mulai);
    ELSE
        SET jumlah_hari = 0;
    END IF;

    -- Menghitung total biaya
    SET NEW.total_biaya = harga_per_hari * jumlah_hari;
END //

DELIMITER ;

INSERT INTO transaksi (id_pelanggan, id_mobil, id_pegawai,
tanggal_mulai, tanggal_selesai, status_transaksi)
VALUES (10, 3, 7, '2024-05-25', '2024-05-28', 'Belum Selesai');
SELECT * FROM transaksi;
SELECT * FROM mobil;



-- NOMOR4--
CREATE TABLE log_hapus_transaksi (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    id_transaksi INT NOT NULL,
    id_pelanggan INT NOT NULL,
    id_mobil INT NOT NULL,
    tanggal_mulai DATE NOT NULL,
    tanggal_selesai DATE NOT NULL,
    total_biaya DECIMAL(10, 2) NOT NULL,
    TIMESTAMP TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
DESC log_hapus_transaksi;
SELECT * FROM log_hapus_transaksi;
DELIMITER //

CREATE TRIGGER after_delete_transaksi
AFTER DELETE ON transaksi
FOR EACH ROW
BEGIN
    INSERT INTO log_hapus_transaksi (id_transaksi, id_pelanggan, id_mobil, tanggal_mulai, tanggal_selesai, total_biaya)
    VALUES (OLD.id_transaksi, OLD.id_pelanggan, OLD.id_mobil, OLD.tanggal_mulai, OLD.tanggal_selesai, OLD.total_biaya);
END //
DELIMITER ;
SELECT * FROM transaksi; -- dapat id 11

-- hapus data tranksasi id 11
DELETE FROM transaksi WHERE id_transaksi = 12;

-- cek apakah trigger berjalan
SELECT * FROM log_hapus_transaksi;
```

4.3 Hasil

1. Soal no 1

id_transaksi	id_pelanggan	id_mobil	id_pegawai	tanggal_mulai	tanggal_Selesai	total_biaya	status_transaksi
1	1	1	1	2024-05-10	2024-05-13	1500000.00	Selesai
2	2	2	2	2024-05-11	2024-05-15	3000000.00	Selesai
3	3	3	3	2024-05-12	2024-05-12	600000.00	Selesai
4	4	4	4	2024-05-13	2024-05-15	1400000.00	Belum Selesai
5	5	5	5	2024-05-14	2023-05-16	1100000.00	Belum Selesai
6	6	6	6	2024-05-15	2023-05-20	3250000.00	Selesai
7	7	7	7	2024-05-16	2023-05-19	2400000.00	Belum Selesai
8	8	8	8	2024-05-17	2023-05-21	3600000.00	Selesai
9	9	9	9	2024-05-19	2023-05-23	2800000.00	Belum Selesai
10	10	10	10	2024-05-22	2023-05-24	2400000.00	Belum Selesai

2. Soal nomor 2

id_log	id_pembayaran	id_transaksi	tanggal_pembayaran	jumlah_pembayaran	metode_pembayaran	TIMESTAMP
1	10	10	2024-05-24	240000.00	Kartu Debit	2024-06-04 17:30:08
2	11	10	2024-05-24	240000.00	Kartu Debit	2024-06-04 17:50:27

3. Soal nomor 3

id_transaksi	id_pelanggan	id_mobil	id_pegawai	tanggal_mulai	tanggal_Selesai	total_biaya	status_transaksi
1	1	1	1	2024-05-10	2024-05-13	1500000.00	Selesai
2	2	2	2	2024-05-11	2024-05-15	3000000.00	Selesai
3	3	3	3	2024-05-12	2024-05-12	600000.00	Selesai
4	4	4	4	2024-05-13	2024-05-15	1400000.00	Belum Selesai
5	5	5	5	2024-05-14	2023-05-16	1100000.00	Selesai
6	6	6	6	2024-05-15	2023-05-20	3250000.00	Selesai
7	7	7	7	2024-05-16	2023-05-19	2400000.00	Belum Selesai
8	8	8	8	2024-05-17	2023-05-21	3600000.00	Selesai
9	9	9	9	2024-05-19	2023-05-23	2800000.00	Belum Selesai
10	10	10	10	2024-05-22	2023-05-24	2400000.00	Belum Selesai

4. Soal nomor 4

id_log	id_transaksi	id_pelanggan	id_mobil	tanggal_mulai	tanggal_selesai	total_biaya	TIMESTAMP
1	11	10	10	2024-05-25	2024-05-28	3600000.00	2024-06-04 17:31:15
2	12	10	3	2024-05-25	2024-05-28	1800000.00	2024-06-04 18:00:57
3	13	10	10	2024-05-25	2024-05-28	3600000.00	2024-06-04 18:24:49

4.4 Penjelasan

Pada praktikum kali ini, praktikan diminta untuk membuat database yang berisikan 6 tabel. Pada tugas paraktikum ini di soal yang pertama praktikan diminta untuk membuat sebuah trigger bernama after_update_transaksi yang akan memperbarui status mobil menjadi tersedia setiap kali status transaksi berubah menjadi selesai. Dan pada nomor berikutnya atau nomor dua, praktikan diminta untuk membuat sebuah trigger bernama after_insert_pembayaran di mana trigger tersebut nantinya akan mencatat log pembayaran ke dalam tabel log_pembayaran setiap kali ada pembayaran baru yang ditambahkan ke tabel pembayaran. Sebelum membuat trigger, praktikan terlebih dahulu diminta untuk membuat tabel log_pembayaran yang nantinya digunakan untuk menyimpan log tersebut.

Lanjut pada soal nomor 3 praktikan diminta untuk membuat sebuah trigger bernama before_insert_transaksi dimana trigger tersebut akan menghitung total biaya sewa secara otomatis sebelum transaksi baru dimasukkan ke dalam table transaksi. Total biaya dihitung berdasarkan harga sewa per hari dari tabel mobil dikalikan dengan jumlah hari sewa (tanggal_selesai - tanggal_mulai). Dan yang terakhir di nomor 4 praktikan diminta untuk membuat trigger bernama after_delete_transaksi dimana trigger tersebut nantinya akan mencatat log setiap kali ada transaksi yang dihapus dari tabel transaksi. Log ini harus mencatat id_transaksi, id_pelanggan, id_mobil, tanggal_mulai, tanggal_selesai, total_biaya, dan timestamp penghapusan. Sama seperti nomor 2, praktian terlebih dahulu diminta untuk membuat tabel log_hapus_transaksi dengan ketentuan yang telah ditetapkan sebelumnya.

BAB V
PENUTUP

5.1 Analisa

Penggunaan trigger dalam MySQL mengubah pendekatan tradisional pasif dalam sistem basis data menjadi lebih aktif, di mana database dapat merespons secara otomatis terhadap operasi tertentu. Dengan demikian, trigger memungkinkan implementasi logika bisnis yang lebih kompleks secara langsung di dalam database, mengurangi kebutuhan akan kode aplikasi tambahan untuk menangani integritas dan konsistensi data. Selain itu, kemampuan trigger untuk otomatisasi proses bisnis dan logging aktivitas pengguna memberikan keuntungan dalam pemantauan dan audit. Namun, ada beberapa keterbatasan, seperti ketidakmampuan untuk memanggil atau membatalkan trigger secara langsung dari program dan pembatasan satu trigger per kombinasi momen dan event per tabel. Hal ini memerlukan perencanaan yang hati-hati dalam desain trigger agar dapat memenuhi semua kebutuhan tanpa konflik. Meskipun demikian, kemampuan trigger untuk menggunakan variabel sistem seperti NEW dan OLD pada operasi UPDATE memberikan fleksibilitas yang signifikan dalam menangani data sebelum dan sesudah perubahan.

5.2 Kesimpulan

1. Trigger memastikan bahwa data yang dimasukkan ke dalam tabel selalu memenuhi aturan dan batasan yang telah ditentukan.
2. Trigger dapat secara otomatis menjalankan logika bisnis tertentu ketika operasi database tertentu terjadi.
3. Trigger mencatat aktivitas pengguna untuk tujuan audit dan pelacakan perubahan data.
4. Trigger tidak dapat dipanggil atau dibatalkan langsung dari program dan hanya mengizinkan satu trigger per kombinasi momen dan event per tabel.
5. Kemampuan untuk menggunakan variabel sistem seperti NEW dan OLD memberikan fleksibilitas tambahan dalam memanipulasi data selama operasi berlangsung.