LAPORAN RESMI
MODUL VII
TRIGGER

NAMA	**:**	**ANISYAFAAH**
N.R.P	**:**	**220441100105**
DOSEN	**:**	**FITRI DAMAYANTI, S.Kom., M.Kom.**
ASISTEN	**:**	**AFFAN MAULANA ZULKARNAIN**
TGL PRAKTIKUM	**:**	**22 MEI 2024**

Disetujui : Mei 2024
Asisten

AFFAN MAULANA ZULKARNAIN
20.04.411.00052

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

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era modern saat ini, kebutuhan akan pengelolaan data yang efisien dan aman menjadi semakin penting seiring dengan pesatnya perkembangan teknologi informasi. Basis data tidak hanya berfungsi sebagai tempat penyimpanan informasi, tetapi juga harus mampu menangani berbagai operasi kompleks dengan cepat dan akurat. Salah satu cara untuk mencapai hal ini adalah dengan menggunakan trigger dalam basis data. Trigger adalah mekanisme otomatis yang dijalankan sebagai respons terhadap peristiwa tertentu dalam basis data, seperti insert, update, atau delete. Dengan memanfaatkan trigger, organisasi dapat memastikan integritas data dan konsistensi operasional tanpa memerlukan intervensi manual yang berlebihan.

Selain menjaga integritas dan konsistensi data, penggunaan trigger dalam basis data juga meningkatkan efisiensi pengolahan data. Di era big data, di mana volume dan kecepatan aliran data sangat tinggi, pengelolaan data secara manual menjadi tidak praktis dan rawan kesalahan. Trigger memungkinkan eksekusi otomatis dari proses-proses penting seperti validasi data, logging, dan auditing. Misalnya, ketika data baru dimasukkan ke dalam sistem, trigger dapat secara otomatis memvalidasi data tersebut berdasarkan aturan-aturan yang telah ditetapkan dan memastikan bahwa data yang masuk memenuhi standar yang diharapkan.

Selain itu, trigger juga memainkan peran penting dalam keamanan data. Dalam lingkungan di mana data seringkali disebarkan dan diakses oleh berbagai pihak, baik internal maupun eksternal, penting untuk memastikan bahwa setiap perubahan pada data tercatat dan diawasi dengan baik. Dengan demikian, penggunaan trigger tidak hanya membantu dalam menjaga kualitas data, tetapi juga memberikan lapisan tambahan perlindungan terhadap ancaman keamanan yang semakin canggih di era digital ini.

1.2 Tujuan

- Mampu mengenal Trigger
- Mampu mendesain trigger sesuai kebutuhan sistem

BAB II
DASAR TEORI

2.1 Database Server

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

Trigger adalah kode perintah SQL yang berisi perintah sql procedural dan perintah deklarativeyang 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 :

- trigger moment (before / after)
- trigger event (insert, delete, update)
- 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 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
```

2 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 datayang baru. Kita bisa menggunakan 2 variabel ini di trigger.

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
```

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 DELETE yang digunakan untuk mengelola data dalam database. Selain itu, dalam konteks pemrograman yang lebih lanjut, kita juga memiliki konsep OLD dan NEW, yang terkait dengan operasi ini. Dalam esai ini, mari kita eksplorasi perbedaan antara INSERT, UPDATE, DELETE, 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 di dalam tabel.
2. Bagaimana penggunaan konsep OLD dan NEW dapat memengaruhi logika bisnis dalam pengembangan perangkat lunak ? Berikan contoh skenario di mana pemahaman tentang OLD dan NEW diperlukan untuk mengimplementasikan logika bisnis yang kompleks.

3.2 Jawab

1. Operasi DELETE digunakan untuk menghapus sebuah baris dari dalam tabel. Berbeda dengan insert dan update yang digunakan untuk menambah data atau baris baru dalam tabel dan untuk mengubah data dalam tabel. Dalam konteks TRIGGER, operasi DELETE biasanya hanya memiliki konsep OLD (nilai data sebelum operasi DML terjadi) karena tidak ada data baru yang dihasilkan, INSERT memiliki konsep NEW (menangani data baru), dan UPDATE memiliki kedua konsep tersebut. Dampak penggunaan operasi DELETE yaitu data yang dihapus tidak lagi tersedia di dalam tabel dan hilang permanen.
2. • OLD : Merujuk pada nilai data sebelum terjadi operasi DML
   • NEW : Merujuk pada nilai data setelah terjadi Operasi DML
   Konsep OLD dan NEW memengaruhi logika bisnis pada pengecekan konsistensi yang memungkinkan pengembang untuk memeriksa perubahan dan memastikan konsistensi data.

Contoh skenario :

Terdapat sebuah tabel karyawan dimana setiap kali data karyawan diperbarui, maka perubahan harus dicatat dalam tabel log.

```
DELIMITER //
CREATE TRIGGER update_employee_log
BEFORE UPDATE ON employee
FOR EACH ROW
  BEGIN
  INSERT INTO employee_log (
  id_employee,
  old_nama,
  new_nama,
  change_date)
    VALUES (
     old.id_employee,
     old.nama,
     new.nama,
     NOW());
END //
DELIMITER ;
```

Kode TRIGGER di atas akan menampilkan catatan perubahan nama yang lama (OLD) dan nama baru (NEW) serta menampilkan tanggal perubahan dan waktu (now()).

BAB IV

IMPLEMENTASI

4.1 Source Code

4.1.1 Mengisi Data Setiap Tabel

- **Tabel Pelanggan**

a) Source Code

```
CREATE DATABASE SewaMobil;

USE SewaMobil;

CREATE TABLE Pelanggan (
        ID_Pelanggan INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
   Nama VARCHAR(100),
   Alamat TEXT,
   No_Telepon VARCHAR(15),
   Email VARCHAR(50)
);

INSERT INTO Pelanggan (ID_Pelanggan, Nama, Alamat, No_Telepon, Email) VALUES
(1, 'Zainal Abidin', 'Jl. Merpati No. 21, Jakarta', '081234567890', 'zainal.abidin@example.com'),
(2, 'Nadia Permata', 'Jl. Cendrawasih No. 12, Bandung', '081234567891', 'nadia.permata@example.com'),
(3, 'Reza Aulia', 'Jl. Rajawali No. 34, Surabaya', '081234567892', 'reza.aulia@example.com'),
(4, 'Siti Rahmawati', 'Jl. Kutilang No. 56, Medan', '081234567893', 'siti.rahmawati@example.com'),
(5, 'Akbar Junaidi', 'Jl. Elang No. 78, Semarang', '081234567894', 'akbar.junaidi@example.com'),
(6, 'Fikri Hakim', 'Jl. Kenari No. 90, Yogyakarta', '081234567895', 'fikri.hakim@example.com'),
(7, 'Rina Kusuma', 'Jl. Merak No. 43, Malang', '081234567896', 'rina.kusuma@example.com'),
(8, 'Galih Saputra', 'Jl. Jalak No. 25, Makassar', '081234567897', 'galih.saputra@example.com'),
```

```
(9, 'Lia Puspita', 'Jl. Kakatua No. 67, Balikpapan', '081234567898', 'lia.puspita@example.com'),
(10, 'Farhan Nugraha', 'Jl. Kepodang No. 89, Palembang', '081234567899', 'farhan.nugraha@example.com');
```

b) Penjelasan

Kode di atas digunakan untuk membuat database Sewa Mobil dan membuat tabel pelanggan serta mengisinya dengan 10 data. Untuk membuat database Sewa Mobil menggunakan perintah CREATE DATABASE Sewa Mobil. Untuk membuat tabel pelanggan menggunakan perintah CREATE TABLE Pelanggan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id pelanggan sebagai primary key, nama, alamat, no telepon, dan email pelanggan. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Tabel Pegawai**

a) Source Code

```
CREATE TABLE Pegawai (
        ID_Pegawai INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
   Nama VARCHAR(100),
   Jabatan VARCHAR(50),
   No_Telepon VARCHAR(15),
   Email VARCHAR(50)
);

INSERT INTO Pegawai (ID_Pegawai, Nama, Jabatan, No_Telepon, Email) VALUES
(1, 'Firdaus Alam', 'Manager', '081234567801', 'firdaus.alam@example.com'),
(2, 'Rifka Dewi', 'Asisten Manager', '081234567802', 'rifka.dewi@example.com'),
(3, 'Hendra Putra', 'Staff Administrasi', '081234567803', 'hendra.putra@example.com'),
```

```
(4, 'Siti Nurhaliza', 'Staff Keuangan', '081234567804', 'siti.nurhaliza@example.com'),
(5, 'Asep Ramadhan', 'Marketing', '081234567805', 'asep.ramadhan@example.com'),
(6, 'Lia Wulandari', 'Customer Service', '081234567806', 'lia.wulandari@example.com'),
(7, 'Rudi Hartono', 'Teknisi', '081234567807', 'rudi.hartono@example.com'),
(8, 'Dian Kartika', 'HRD', '081234567808', 'dian.kartika@example.com'),
(9, 'Eka Pratama', 'Security', '081234567809', 'eka.pratama@example.com'),
(10, 'Gina Setiawati', 'Cleaning Service', '081234567810', 'gina.setiawati@example.com');
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel pegawai dan mengisi data sebanyak 10 data. Untuk membuat tabel pegawai menggunakan perintah CREATE TABLE Pegawai (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id pegawai sebagai primary key, nama, jabatan, no telepon, dan email pegawai. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Tabel Mobil**

a) Source Code

```
CREATE TABLE Mobil (
        ID_Mobil INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
   Merk VARCHAR(50),
   Model VARCHAR(50),
   Tahun INT(11),
   Warna VARCHAR(20),
   Harga_Sewa DECIMAL(10.2),
   Status ENUM("Tersedia", "Tidak Tersedia")
);
```

```
INSERT INTO Mobil (ID_Mobil, Merk, Model, Tahun, Warna, Harga_Sewa, Status) VALUES
(1, 'Toyota', 'Avanza', 2020, 'Putih', 350000.00, 'Tidak Tersedia'),
(2, 'Honda', 'Civic', 2019, 'Hitam', 450000.00, 'Tersedia'),
(3, 'Suzuki', 'Ertiga', 2018, 'Merah', 300000.00, 'Tidak Tersedia'),
(4, 'Mitsubishi', 'Pajero', 2021, 'Silver', 600000.00, 'Tidak Tersedia'),
(5, 'Daihatsu', 'Xenia', 2020, 'Biru', 320000.00, 'Tersedia'),
(6, 'Nissan', 'Juke', 2017, 'Kuning', 400000.00, 'Tidak Tersedia'),
(7, 'Kia', 'Rio', 2019, 'Hijau', 280000.00, 'Tersedia'),
(8, 'Hyundai', 'Tucson', 2021, 'Abu-Abu', 550000.00, 'Tersedia'),
(9, 'Ford', 'Everest', 2020, 'Coklat', 480000.00, 'Tersedia'),
(10, 'Chevrolet', 'Trailblazer', 2018, 'Oranye', 470000.00, 'Tidak Tersedia');
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel mobil dan mengisi data sebanyak 10 data. Untuk membuat tabel mobil menggunakan perintah CREATE TABLE Mobil (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id mobil sebagai primary key, merk, model, tahun, nama, harga sewa, dan status. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Tabel Perawatan**

a) Source Code

```
CREATE TABLE Perawatan (
        ID_Perawatan INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ID_Mobil INT(11),
    Tanggal DATE,
    Deskripsi TEXT,
    Biaya DECIMAL,
    FOREIGN KEY (ID_Mobil) REFERENCES Mobil (ID_Mobil)
);
```

```
INSERT INTO Perawatan (ID_Perawatan, ID_Mobil, Tanggal, Deskripsi, Biaya) VALUES
(1, 1, '2024-04-01', 'Ganti oli mesin', 300000.00),
(2, 2, '2024-04-02', 'Perbaikan rem', 500000.00),
(3, 3, '2024-04-03', 'Penggantian ban', 800000.00),
(4, 4, '2024-04-04', 'Servis rutin', 600000.00),
(5, 5, '2024-04-05', 'Ganti filter udara', 200000.00),
(6, 6, '2024-04-06', 'Penggantian aki', 700000.00),
(7, 7, '2024-04-07', 'Perbaikan AC', 400000.00),
(8, 8, '2024-04-08', 'Penggantian lampu depan', 150000.00),
(9, 9, '2024-04-09', 'Perbaikan transmisi', 1200000.00),
(10, 10, '2024-04-10', 'Penggantian wiper', 100000.00);
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel perawatan dan mengisi data sebanyak 10 data. Untuk membuat tabel perawatan menggunakan perintah CREATE TABLE Perawatan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id perawatan sebagai primary key, id_mobil yang terhubung dengan tabel mobil menggunakan FOREIGN KEY, tanggal, deskripsi, dan biaya. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Tabel Transaksi**

a) Source Code

```
CREATE TABLE Transaksi (
        ID_Transaksi INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
   ID_Pelanggan INT(11),
   ID_Mobil INT(11),
   ID_Pegawai INT(11),
   Tanggal_Mulai DATE,
   Tanggal_Selesai DATE,
   Total_Biaya DECIMAL(10.2),
```

```
    Status_Transaksi ENUM("Selesai", "Belum Selesai"),
    FOREIGN KEY (ID_Pelanggan) REFERENCES Pelanggan (ID_Pelanggan),
    FOREIGN KEY (ID_Mobil) REFERENCES Mobil (ID_Mobil),
    FOREIGN KEY (ID_Pegawai) REFERENCES Pegawai (ID_Pegawai)
);

INSERT INTO Transaksi (ID_Transaksi, ID_Pelanggan, ID_Mobil, ID_Pegawai, Tanggal_Mulai, Tanggal_Selesai, Total_Biaya, Status_Transaksi) VALUES
(1, 1, 1, 1, '2024-05-01', '2024-05-05', 1400000.00, 'Selesai'),
(2, 2, 2, 2, '2024-05-02', '2024-05-06', 1800000.00, 'Selesai'),
(3, 3, 3, 3, '2024-05-03', '2024-05-07', 1200000.00, 'Selesai'),
(4, 4, 4, 4, '2024-05-04', '2024-05-08', 2400000.00, 'Selesai'),
(5, 5, 5, 5, '2024-05-05', '2024-05-09', 1280000.00, 'Selesai'),
(6, 6, 6, 6, '2024-05-06', '2024-05-10', 1600000.00, 'Selesai'),
(7, 7, 7, 7, '2024-05-07', '2024-05-11', 1120000.00, 'Belum Selesai'),
(8, 8, 8, 8, '2024-05-08', '2024-05-12', 2200000.00, 'Belum Selesai'),
(9, 9, 9, 9, '2024-05-09', '2024-05-13', 1920000.00, 'Belum Selesai'),
(10, 10, 10, 10, '2024-05-10', '2024-05-14', 1880000.00, 'Belum Selesai');
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel transaksi dan mengisi data sebanyak 10 data. Untuk membuat tabel transaksi menggunakan perintah CREATE TABLE Transaksi (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id transaksi sebagai primary key, id pelanggan yang terhubung dengan tabel pelanggan, id mobil yang terhubung dengan tabel mobil, idd pegawai yang terhubung dengan tabel pegawai dengan FOREIGN KEY, tanggal mulai, tanggal selesai, total biaya, dan status. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Tabel Pembayaran**

a) Source Code

```
CREATE TABLE Pembayaran (
        ID_Pembayaran INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
   ID_Transaksi INT(11),
   Tanggal_Pembayaran DATE,
   Jumlah_Pembayaran DECIMAL(10.2),
   Metode_Pembayaran VARCHAR(50),
   FOREIGN KEY (ID_Transaksi) REFERENCES Transaksi (ID_Transaksi)
);

INSERT INTO Pembayaran (ID_Pembayaran, ID_Transaksi, Tanggal_Pembayaran, Jumlah_Pembayaran, Metode_Pembayaran) VALUES
(1, 1, '2024-05-05', 1400000.00, 'Transfer Bank'),
(2, 2, '2024-05-06', 1800000.00, 'Kartu Kredit'),
(3, 3, '2024-05-07', 1200000.00, 'Transfer Bank'),
(4, 4, '2024-05-08', 2400000.00, 'Kartu Kredit'),
(5, 5, '2024-05-09', 1280000.00, 'Transfer Bank'),
(6, 6, '2024-05-10', 1600000.00, 'Kartu Kredit'),
(7, 7, '2024-05-11', 1120000.00, 'Transfer Bank'),
(8, 8, '2024-05-12', 2200000.00, 'Kartu Kredit'),
(9, 9, '2024-05-13', 1920000.00, 'Transfer Bank'),
                b)  (10, 10, '2024-05-14', 1880000.00, 'Kartu Kredit');
```

c) Penjelasan

Kode di atas digunakan untuk membuat tabel pembayaran dan mengisi data sebanyak 10 data. Untuk membuat tabel pembayaran menggunakan perintah CREATE TABLE Pembayaran (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id pembayaran sebagai primary key, id transaksi ynag terhubung dengan tabel transaksi dengan FOREIGN KEY, tanggal pembayaran, jumlah pembayaran, dan metode pembayaran Selanjutnya untuk

mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

4.1.2 Trigger Untuk Memperbarui Status Mobil Menjadi Tersedia Setiap Status Transaksi Selesai

a) Source Code

```
-- Nomor 1
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
= 1;
SELECT * FROM Mobil;
```

b) Penjelasan

Kode di atas digunakan untuk membuat sebuah trigger untuk mengubah status mobil. Trigger ini menggunakan AFTER UPDATE pada tabel transaksi yaitu status transaksi yang baru diset selesai ketika tabel status mobil tersedia. Perubahan ini didasarkan pada id mobil sehingga ketika mengupdate status transaksi selesai pada tabel transaksi dan id mobil 1, program akan mengubah status id mobil 1 menjadi selesai.

4.1.3 Trigger Untuk Mencatat Log Pembayaran Setiap Ada Pembayaran Baru Pada Tabel Pembayaran

a) Source Code

```
-- Nomor 2
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
);

DELIMITER //
CREATE TRIGGER After_Insert_Pembayaran
AFTER INSERT ON Pembayaran
FOR EACH ROW
BEGIN
 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.Jumlah_Pembayaran,
NEW.Metode_Pembayaran);
END //
DELIMITER ;

INSERT INTO Pembayaran (ID_Transaksi, Tanggal_Pembayaran,
Jumlah_Pembayaran, Metode_Pembayaran)
VALUES (5, '2024-05-09', 1280000.00, 'Tunai');
SELECT * FROM Pembayaran;
SELECT * FROM Log_Pembayaran;
```

b) Penjelasan

Kode di atas digunakan untuk menambah tabel Log Pembayaran yang berisi id log, id pembayaran, id transaksi, tanggal pembayaran, jumlah pembayaran, metode pembayaran, dan timstamp (diatur

defaultnya untuk tanggal dan waktu hari ini). Kemudian membuat trigger AFTER INSERT pada tabel pembayaran dimana data baru akan dimasukkan dalam tabel log pembayaran dengan semua kolom sehingga menggunakan perintah NEW. Sehingga setiap kali akan memasukkan data baru pada tabel pembayaran, data tersebut juga akan masuk pada tabel log pembayaran.

4.1.4 Trigger Untuk Menghitung Total Biaya Sewa Sebelum Transaksi Baru Dimasukkan

a) Source Code

```
-- Nomor 3
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;
   SELECT Harga_Sewa INTO Harga_Per_Hari
   FROM Mobil
   WHERE ID_Mobil = NEW.ID_Mobil;
   SET Jumlah_Hari = DATEDIFF(NEW.Tanggal_Selesai,
NEW.Tanggal_Mulai);
   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 (9, 6, 1, '2024-06-06', '2024-06-08', 'Selesai');
INSERT INTO Transaksi (ID_Pelanggan, ID_Mobil, ID_Pegawai,
Tanggal_Mulai, Tanggal_Selesai, Status_Transaksi)
VALUES (4, 1, 3, '2024-06-15', '2024-06-21', 'Selesai');
SELECT * FROM Transaksi;
```

b) Penjelasan

Kode di atas digunakan untuk menambah data dimana total biaya akan dihitung berdasarkan total berapa hari mobil dipinjam. Kode harus mendeklarasikan dua variabel untuk harga per hari dan jumlah hari. Kemudian tampilkan harga sewa dari tabel mobil berdasarkan id mobil yang baru dan set variabel jumlah hari dengan pengurangan antara tanggal selesai yang baru dengan tanggal mulai yang baru serta set biaya yang baru dengan harga per hari dikali jumlah harinya. Sehingga ketika menambah data baru, total biayanya akan dihitung berdasarkan banyak hari.

4.1.5 Trigger Untuk Mencatat Log Setiap Ada Transaksi Yang Dihapus

a) Source Code

```
-- Nomor 4
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
);

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 ;
```

```
DELETE FROM Transaksi WHERE ID_Transaksi = 12;
SELECT * FROM Transaksi;
SELECT * FROM Log_Hapus_Transaksi;
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel log transaksi yang terdiri dari id log, id transaksi, id pelanggan, id mobil, tanggal mulai, tanggal selesai, total biaya, dan timestamp (diatur defaultnya untuk tanggal dan waktu hari ini) serta membuat trigger AFTER DELETE untuk menambah data lama ke tabel log transaksi. Trigger hanya berisi kode untuk memasukkan data lama yang akan dihapus pada tabel log hapus transaksi menggunakan peintah OLD. Sehingga setiap kali ingin menghapus data pada tabel transaksi berdasarkan id, data lama tersebut akan masuk ke dalam tabel log hapus transaksi.

4.2 Hasil

4.2.1 Data Setiap Tabel

- **Tabel Pelanggan**

ID_Pelanggan	Nama	Alamat	No_Telepon	Email
1	Zainal Abidin	Jl. Merpati No. 21, Jakarta	081234567890	zainal.abidin@example.com
2	Nadia Permata	Jl. Cendrawasih No. 12, Bandung	081234567891	nadia.permata@example.com
3	Reza Aulia	Jl. Rajawali No. 34, Surabaya	081234567892	reza.aulia@example.com
4	Siti Rahmawati	Jl. Kutilang No. 56, Medan	081234567893	siti.rahmawati@example.com
5	Akbar Junaidi	Jl. Elang No. 78, Semarang	081234567894	akbar.junaidi@example.com
6	Fikri Hakim	Jl. Kenari No. 90, Yogyakarta	081234567895	fikri.hakim@example.com
7	Rina Kusuma	Jl. Merak No. 43, Malang	081234567896	rina.kusuma@example.com
8	Galih Saputra	Jl. Jalak No. 25, Makassar	081234567897	galih.saputra@example.com
9	Lia Puspita	Jl. Kakatua No. 67, Balikpapan	081234567898	lia.puspita@example.com
10	Farhan Nugraha	Jl. Kepodang No. 89, Palembang	081234567899	farhan.nugraha@example.com

- **Tabel Pegawai**

ID_Pegawai	Nama	Jabatan	No_Telepon	Email
1	Firdaus Alam	Manager	081234567801	firdaus.alam@example.com
2	Rifka Dewi	Asisten Manager	081234567802	rifka.dewi@example.com
3	Hendra Putra	Staff Administrasi	081234567803	hendra.putra@example.com
4	Siti Nurhaliza	Staff Keuangan	081234567804	siti.nurhaliza@example.com
5	Asep Ramadhan	Marketing	081234567805	asep.ramadhan@example.com
6	Lia Wulandari	Customer Service	081234567806	lia.wulandari@example.com
7	Rudi Hartono	Teknisi	081234567807	rudi.hartono@example.com
8	Dian Kartika	HRD	081234567808	dian.kartika@example.com
9	Eka Pratama	Security	081234567809	eka.pratama@example.com
10	Gina Setiawati	Cleaning Service	081234567810	gina.setiawati@example.com

- **Tabel Mobil**

ID_Mobil	Merk	Model	Tahun	Warna	Harga_Sewa	Status
1	Toyota	Avanza	2020	Putih	350000	Tidak Tersedia
2	Honda	Civic	2019	Hitam	450000	Tersedia
3	Suzuki	Ertiga	2018	Merah	300000	Tidak Tersedia
4	Mitsubishi	Pajero	2021	Silver	600000	Tidak Tersedia
5	Daihatsu	Xenia	2020	Biru	320000	Tersedia
6	Nissan	Juke	2017	Kuning	400000	Tidak Tersedia
7	Kia	Rio	2019	Hijau	280000	Tersedia
8	Hyundai	Tucson	2021	Abu-Abu	550000	Tersedia
9	Ford	Everest	2020	Coklat	480000	Tersedia
10	Chevrolet	Trailblazer	2018	Oranye	470000	Tidak Tersedia

- **Tabel Perawatan**

ID_Perawatan	ID_Mobil	Tanggal	Deskripsi	Biaya
1	1	2024-04-01	Ganti oli mesin	300000
2	2	2024-04-02	Perbaikan rem	500000
3	3	2024-04-03	Penggantian ban	800000
4	4	2024-04-04	Servis rutin	600000
5	5	2024-04-05	Ganti filter udara	200000
6	6	2024-04-06	Penggantian aki	700000
7	7	2024-04-07	Perbaikan AC	400000
8	8	2024-04-08	Penggantian lampu depan	150000
9	9	2024-04-09	Perbaikan transmisi	1200000
10	10	2024-04-10	Penggantian wiper	100000

- **Tabel Transaksi**

ID_Transaksi	ID_Pelanggan	ID_Mobil	ID_Pegawai	Tanggal_Mulai	Tanggal_Selesai	Total_Biaya	Status_Transaksi
1	1	1	1	2024-05-01	2024-05-05	1400000	Selesai
2	2	2	2	2024-05-02	2024-05-06	1800000	Selesai
3	3	3	3	2024-05-03	2024-05-07	1200000	Selesai
4	4	4	4	2024-05-04	2024-05-08	2400000	Selesai
5	5	5	5	2024-05-05	2024-05-09	1280000	Selesai
6	6	6	6	2024-05-06	2024-05-10	1600000	Selesai
7	7	7	7	2024-05-07	2024-05-11	1120000	Belum Selesai
8	8	8	8	2024-05-08	2024-05-12	2200000	Belum Selesai
9	9	9	9	2024-05-09	2024-05-13	1920000	Belum Selesai
10	10	10	10	2024-05-10	2024-05-14	1880000	Belum Selesai

- **Tabel Pembayaran**

ID_Pembayaran	ID_Transaksi	Tanggal_Pembayaran	Jumlah_Pembayaran	Metode_Pembayaran
1	1	2024-05-05	1400000	Transfer Bank
2	2	2024-05-06	1800000	Kartu Kredit
3	3	2024-05-07	1200000	Transfer Bank
4	4	2024-05-08	2400000	Kartu Kredit
5	5	2024-05-09	1280000	Transfer Bank
6	6	2024-05-10	1600000	Kartu Kredit
7	7	2024-05-11	1120000	Transfer Bank
8	8	2024-05-12	2200000	Kartu Kredit
9	9	2024-05-13	1920000	Transfer Bank
10	10	2024-05-14	1880000	Kartu Kredit

4.2.2 Trigger Untuk Memperbarui Status Mobil Menjadi Tersedia Setiap Status Transaksi Selesai

ID_Mobil	Merk	Model	Tahun	Warna	Harga_Sewa	Status
1	Toyota	Avanza	2020	Putih	350000	Tersedia
2	Honda	Civic	2019	Hitam	450000	Tersedia
3	Suzuki	Ertiga	2018	Merah	300000	Tidak Tersedia
4	Mitsubishi	Pajero	2021	Silver	600000	Tidak Tersedia
5	Daihatsu	Xenia	2020	Biru	320000	Tersedia
6	Nissan	Juke	2017	Kuning	400000	Tidak Tersedia
7	Kia	Rio	2019	Hijau	280000	Tersedia
8	Hyundai	Tucson	2021	Abu-Abu	550000	Tersedia
9	Ford	Everest	2020	Coklat	480000	Tersedia
10	Chevrolet	Trailblazer	2018	Oranye	470000	Tidak Tersedia

4.2.3 Trigger Untuk Mencatat Log Pembayaran Setiap Ada Pembayaran Baru Pada Tabel Pembayaran

ID_Pembayaran	ID_Transaksi	Tanggal_Pembayaran	Jumlah_Pembayaran	Metode_Pembayaran
1	1	2024-05-05	1400000	Transfer Bank
2	2	2024-05-06	1800000	Kartu Kredit
3	3	2024-05-07	1200000	Transfer Bank
4	4	2024-05-08	2400000	Kartu Kredit
5	5	2024-05-09	1280000	Transfer Bank
6	6	2024-05-10	1600000	Kartu Kredit
7	7	2024-05-11	1120000	Transfer Bank
8	8	2024-05-12	2200000	Kartu Kredit
9	9	2024-05-13	1920000	Transfer Bank
10	10	2024-05-14	1880000	Kartu Kredit
11	5	2024-05-09	1280000	Tunai

ID_Log	ID_Pembayaran	ID_Transaksi	Tanggal_Pembayaran	Jumlah_Pembayaran	Metode_Pembayaran	timestamp
1	11	5	2024-05-09	1280000.00	Tunai	2024-05-30 15:28:47

4.2.4 Trigger Untuk Menghitung Total Biaya Sewa Sebelum Transaksi Baru Dimasukkan

ID_Transaksi	ID_Pelanggan	ID_Mobil	ID_Pegawai	Tanggal_Mulai	Tanggal_Selesai	Total_Biaya	Status_Transaksi
1	1	1	1	2024-05-01	2024-05-05	1400000	Selesai
2	2	2	2	2024-05-02	2024-05-06	1800000	Selesai
3	3	3	3	2024-05-03	2024-05-07	1200000	Selesai
4	4	4	4	2024-05-04	2024-05-08	2400000	Selesai
5	5	5	5	2024-05-05	2024-05-09	1280000	Selesai
6	6	6	6	2024-05-06	2024-05-10	1600000	Selesai
7	7	7	7	2024-05-07	2024-05-11	1120000	Belum Selesai
8	8	8	8	2024-05-08	2024-05-12	2200000	Belum Selesai
9	9	9	9	2024-05-09	2024-05-13	1920000	Belum Selesai
10	10	10	10	2024-05-10	2024-05-14	1880000	Belum Selesai
11	9	6	1	2024-06-06	2024-06-08	800000	Selesai
12	4	1	3	2024-06-15	2024-06-21	2100000	Selesai

4.2.5 Trigger Untuk Mencatat Log Setiap Ada Transaksi Yang Dihapus

ID_Transaksi	ID_Pelanggan	ID_Mobil	ID_Pegawai	Tanggal_Mulai	Tanggal_Selesai	Total_Biaya	Status_Transaksi
1	1	1	1	2024-05-01	2024-05-05	1400000	Selesai
2	2	2	2	2024-05-02	2024-05-06	1800000	Selesai
3	3	3	3	2024-05-03	2024-05-07	1200000	Selesai
4	4	4	4	2024-05-04	2024-05-08	2400000	Selesai
5	5	5	5	2024-05-05	2024-05-09	1280000	Selesai
6	6	6	6	2024-05-06	2024-05-10	1600000	Selesai
7	7	7	7	2024-05-07	2024-05-11	1120000	Belum Selesai
8	8	8	8	2024-05-08	2024-05-12	2200000	Belum Selesai
9	9	9	9	2024-05-09	2024-05-13	1920000	Belum Selesai
10	10	10	10	2024-05-10	2024-05-14	1880000	Belum Selesai
11	9	6	1	2024-06-06	2024-06-08	800000	Selesai

ID_Log	ID_Transaksi	ID_Pelanggan	ID_Mobil	Tanggal_Mulai	Tanggal_Selesai	Total_Biaya	timestamp
1	12	4	1	2024-06-15	2024-06-21	2100000.00	2024-05-30 15:48:36

BAB V
PENUTUP

5.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan trigger dalam basis data menawarkan sejumlah keuntungan signifikan yang dapat meningkatkan efisiensi dan keamanan pengelolaan data. Dengan trigger, operasi seperti validasi data, pelacakan perubahan, dan pemeliharaan integritas referensial dapat dilakukan secara otomatis tanpa memerlukan campur tangan manual. Dalam konteks aplikasi bisnis, trigger dapat mengotomatisasi berbagai alur kerja, seperti pengiriman notifikasi saat terjadi perubahan penting dalam data, yang dapat meningkatkan responsivitas dan layanan kepada pengguna.

Namun, meskipun memiliki banyak manfaat, penggunaan trigger juga datang dengan beberapa tantangan dan potensi kelemahan yang perlu diperhatikan. Salah satu masalah utama adalah kompleksitas pengelolaan dan pemeliharaan trigger, terutama dalam sistem basis data besar dengan banyak trigger yang saling terkait. Kesalahan dalam penulisan atau pengelolaan trigger dapat menyebabkan masalah performa yang signifikan, seperti deadlocks atau penurunan kecepatan transaksi. Selain itu, karena trigger berjalan di tingkat basis data, mereka dapat sulit dilacak dan di-debug dibandingkan dengan kode aplikasi biasa.

5.2 Kesimpulan

1. 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
2. MySQL akan menjalankan trigger ketika ada user, atau store procedur yang menjalankan perintah database tertentu, yaitu ketika menambahkan baris data ke tabel atauketika menghapus semua data dari tabel.
3. 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 datayang baru.