KLINIK BINA SEHAT

2025

Banu Najieh Al-Fadhil
2400018060

Klinik Binasehat

(Implementasi Basis Data Relasional Menggunakan MySQL)

Identitas Penulis:

Nama : Banu Najieh Al-Fadhil

NIM : 2400018060

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Klinik Binasehat saat ini masih menggunakan sistem pencatatan manual untuk mengelola data dokter, pasien, riwayat pemeriksaan, hingga transaksi obat. Pencatatan yang tidak berbasis komputer tersebut menimbulkan berbagai kendala, terutama pada integrasi data antar bagian. Kondisi ini menjadikan proses pelayanan menjadi kurang efektif dan memperbesar potensi kesalahan pencatatan.

Selain itu, tidak adanya sistem basis data membuat proses pencarian informasi seperti data pasien, riwayat pemeriksaan, ataupun transaksi obat membutuhkan waktu lebih lama. Hal ini berdampak pada lambatnya pelayanan dan kurang optimalnya pengambilan keputusan administratif maupun medis. Oleh karena itu, diperlukan sistem berbasis basis data yang mampu menyimpan, mengolah, serta menghubungkan data secara terstruktur, cepat, dan akurat.

1.2 Rumusan Masalah

Berdasarkan kondisi dan latar belakang tersebut, maka dapat dirumuskan beberapa masalah sebagai berikut:

1. Data tidak terintegrasi antara bagian dokter, pasien, dan administrasi.
2. Risiko kehilangan atau kerusakan data tinggi karena sistem masih berbasis dokumen manual.
3. Proses pencarian data pasien dan riwayat pemeriksaan membutuhkan waktu lama.
4. Tidak adanya sistem query untuk mendukung pencarian dan pelaporan data secara cepat.
5. Penghitungan stok obat dan total transaksi masih dilakukan manual sehingga rawan kesalahan.
6. Keamanan dan kerahasiaan data medis belum terjamin karena tidak ada akses kontrol pengguna.

1.3 Tujuan

Tujuan dari perancangan sistem basis data Klinik Binasehat ini adalah untuk:

1. Membangun penyimpanan data yang terstruktur dan terintegrasi antar bagian.
2. Mempermudah pencarian dan pelaporan data medis maupun transaksi.
3. Meningkatkan keakuratan pengelolaan stok obat dan transaksi keuangan.

4. Mengurangi risiko kesalahan pencatatan melalui validasi di dalam sistem.
5. Menyediakan data yang cepat diakses untuk membantu proses pengambilan keputusan.
6. Meningkatkan keamanan data melalui sistem hak akses pengguna yang jelas.

1.4 Batasan Masalah

Agar pembahasan tidak terlalu luas, maka batasan masalah dalam laporan ini adalah:

1. Sistem hanya berfokus pada pengelolaan dokter, pasien, riwayat pemeriksaan, obat, dan transaksi.
2. Pembahasan hanya pada perancangan basis data, belum termasuk implementasi penuh aplikasi klinik.
3. Pengelolaan stok obat terbatas pada pencatatan jumlah masuk, keluar, dan transaksi pembelian pasien.
4. Sistem keamanan dibatasi pada pemberian hak akses untuk admin, dokter, dan administrasi.
5. Tidak membahas integrasi dengan rumah sakit lain atau sistem kesehatan pihak ketiga.

BAB II
ANALISIS MASALAH DAN KEBUTUHAN SISTEM

2.1 Deskripsi Sistem

Sistem basis data Klinik Binasehat dirancang untuk membantu pengelolaan informasi terkait dokter, pasien, riwayat pemeriksaan, obat, serta transaksi penjualan obat. Sistem ini berfungsi sebagai pusat penyimpanan data yang terintegrasi, sehingga setiap proses pelayanan medis dan apotek dapat saling berhubungan dengan jelas.

Database yang dirancang bertujuan untuk:

- Menghubungkan data antar entitas (dokter, pasien, obat, dan transaksi).
- Menyediakan akses data secara cepat melalui query SQL.
- Menjamin keamanan dan konsistensi data melalui aturan relasi dan *constraint*.
- Mempermudah proses pencarian data medis maupun transaksi keuangan secara akurat.

Dengan adanya sistem basis data ini, proses pelayanan klinik menjadi lebih terstruktur, efisien, dan minim risiko kesalahan pencatatan.

2.2 Identifikasi Masalah Sistem

Berdasarkan kondisi sistem manual yang berjalan, diperoleh beberapa masalah utama sebagai berikut:

1. Data dokter, pasien, dan obat belum terintegrasi sehingga sulit dilacak keterkaitannya.
2. Berkas data mudah hilang atau rusak karena penyimpanan masih manual.
3. Proses pencarian riwayat pemeriksaan pasien membutuhkan waktu lama.
4. Tidak ada sistem yang menghitung stok obat secara otomatis saat terjadi transaksi.
5. Kesalahan input transaksi sering terjadi karena perhitungan masih manual.
6. Tidak adanya kontrol hak akses menyebabkan keamanan data medis kurang terjamin.

2.3 Pemetaan Masalah ke Solusi SQL

Berikut pemetaan setiap masalah ke solusi yang dapat diimplementasikan melalui basis data dan perintah SQL:

1. Data tidak terintegrasi: Membuat relasi antar tabel dengan FOREIGN KEY (Doctor–History, Patient–History, dll).
2. Risiko kehilangan berkas: Membuat penyimpanan terpusat menggunakan Database Management System (DBMS).
3. Pencarian data lama: Menggunakan SELECT + WHERE untuk pencarian cepat.

4. Stok obat tidak terkontrol: Implementasi TRIGGER untuk mengurangi stok otomatis saat transaksi insert.
5. Kesalahan perhitungan transaksi: Menggunakan Agregat dengan SUM dan COUNT.
6. Tidak ada keamanan akses: Penerapan role user & GRANT/REVOKE untuk membatasi hak akses.

BAB III
PERANCANGAN BASIS DATA (ERD)

3.1 Tujuan Perancangan ERD

Perancangan Entity Relationship Diagram (ERD) bertujuan untuk menggambarkan struktur data dan hubungan antar entitas dalam sistem secara jelas dan terorganisir. Dengan adanya ERD, setiap data seperti dokter, pasien, riwayat pemeriksaan, obat, hingga transaksi dapat dimodelkan sesuai fungsi dan keterkaitannya, sehingga alur informasi di dalam sistem lebih mudah dipahami. Desain ini juga membantu mencegah terjadinya duplikasi, inkonsistensi, dan kesalahan integritas data karena setiap entitas, atribut, dan relasi sudah ditentukan sejak awal sebelum database dibuat. Selain itu, ERD menjadi dasar atau blueprint dalam proses implementasi tabel, penentuan primary key, foreign key, serta aturan relasi di dalam SQL, sehingga proses pembangunan sistem berlangsung lebih terarah dan efisien. Singkatnya, ERD dibuat untuk memastikan sistem basis data Klinik Binasehat dapat berjalan konsisten, terstruktur, mudah diakses, dan mendukung kebutuhan pengolahan informasi secara akurat.

3.2 Identifikasi Entitas dan Jenisnya

Entitas	Tipe	Keterangan
Pasien	**Kuat**	Berdiri sendiri
Dokter	**Kuat**	Berdiri sendiri
Perawat	**Kuat**	Berdiri sendiri
Obat	**Kuat**	Berdiri sendiri
Rawat_Inap	**Kuat**	Identitas mandiri
Pembayaran	**Kuat**	Memiliki identitas sendiri
Rekam_Medis	Lemah	Bergantung pada pasien/dokter/rawat inap
Detail_Rawat_Inap	Lemah	Bergantung pada rawat inap
Detail_Obat	Lemah	Bergantung pada pembayaran/obat

3.3 Identifikasi Atribut dan Jenis Atribut

Dari keterangan diatas didapai beberapa entitas yang berkaitan beserta atributnya, diantaranya:

1. Pasien → (id_pasien (PK), nama_pasien, nik, jenis_kelamin, alamat, no_telp, gol_darah)
2. Dokter → (id_dokter (PK), nama_dokter, spesialisasi, no_telp)
3. Perawat → (id_perawat (PK), nama, no_telp)
4. Rekam_Medis → (id_rekam (PK), id_pasien (FK), id_dokter (FK), keluhan, diagnosa, tindakan, tanggal)

5. Rawat_Inap -> (id_rawat_inap (PK), kamar, kelas, tanggal_masuk, tanggal_keluar, status_rawat)
6. Detail_Rawat_Inap -> (id_detail_rawat (PK), id_rawat (FK), id_dokter (FK), id_perawat (FK), tgl_mulai, tgl_selesai, catatan
7. Pembayaran → (id_pembayaran (PK), id_pasien (FK), id_rekam (FK), total_biaya, status)
8. Detail_Pembayaran → (id_detail (PK), id_pembayaran (FK), id_obat (FK), jumlah, subtotal)
9. Obat → (id_obat (PK), nama_obat, jenis, harga_satuan)

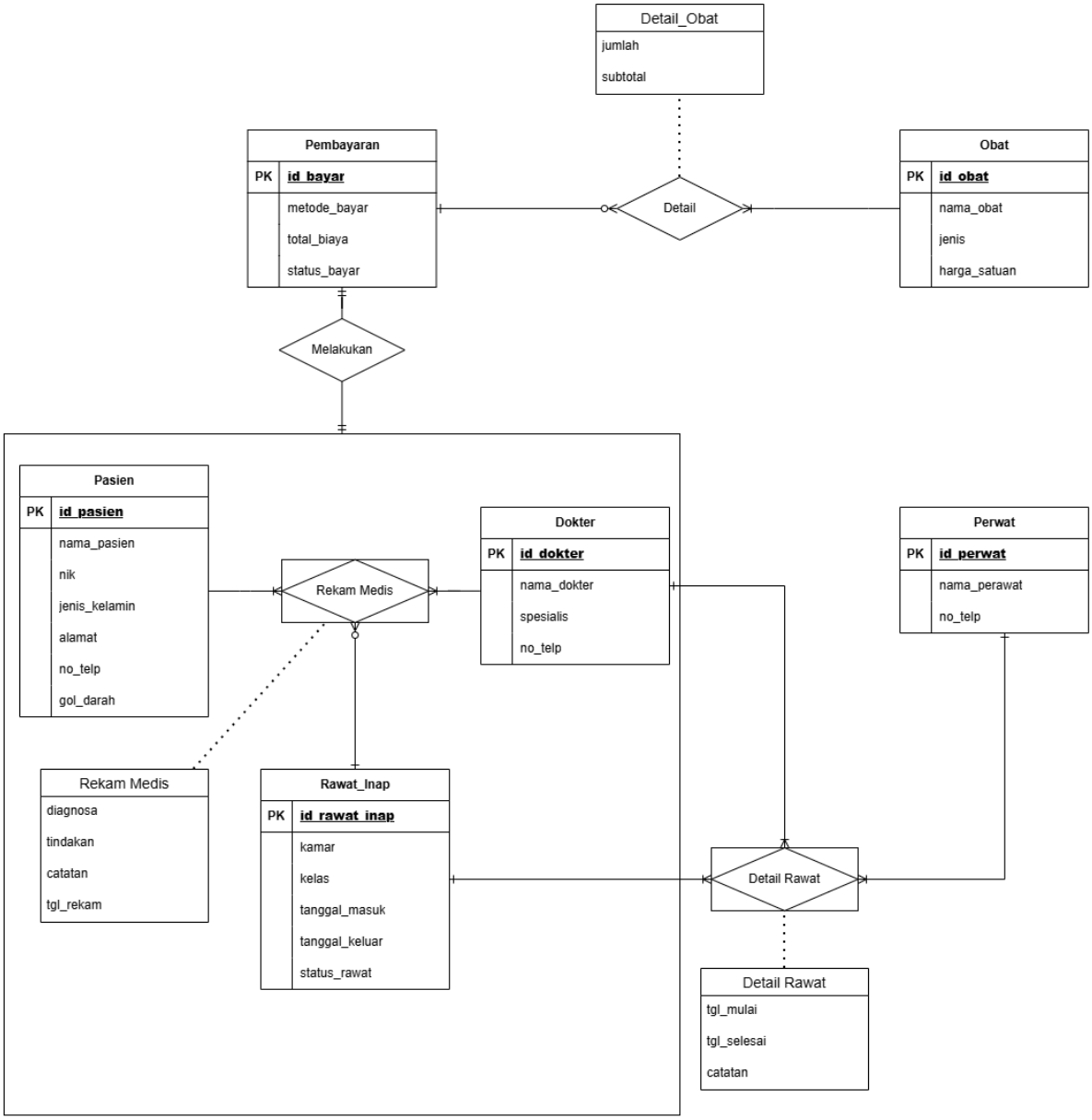
3.4 Relasi dan Kardinalitas

Entitas	Kardinalitas	Derajat (Min,Max)	jenis Relasi	Keterangan
Pasien — Rekam_Medis	1 : M	(1, N)	One to Many	1 Pasien memiliki banyak Rekam Medis
Dokter — Rekam_Medis	1 : M	(1, N)	One to Many	1 Dokter menangani banyak Rekam Medis
Rawat_Inap — Rekam_Medis	1 : M	(1, 0/N)	One to Optional Many	1 Rawat Inap bisa memiliki banyak rekam medis selama masa rawat
Rawat_Inap — Detail_Rawat_Inap	1 : M	(1, N)	One Mandatory to Many Mandatory	1 Rawat Inap memiliki banyak detail aktivitas rawat (per dokter/perawat)
Dokter — Detail_Rawat_Inap	1 : M	(1, N)	One to Many Mandatory	1 Dokter bisa menangani banyak detail rawat inap.
Perawat — Detail_Rawat_Inap	1 : M	(1, N)	One to Many Mandatory	1 Perawat bisa menangani banyak detail rawat inap.
Rekam_Medis — Pembayaran	1 : 1 (opsional)	(1, 1)	One Mandatory to One Mandatory	1 Rekam Medis wajib memiliki satu Pemabayarn
Pembayaran — Detail_Obat	1 : M	(1, 0/N)	One to Many Optional	1 Pembayaran bisa memiliki banyak detail obat atau tidak
Obat — Detail_Obat	1 : M	(1, N)	One to Many	1 Obat dapat muncul di banyak detail obat

Relasi	Entitas 1	Partisipasi	Entitas 2	Partisipasi
Pasien — Rekam_Medis	Pasien	Total	Rekam_Medis	Total
Dokter — Rekam_Medis	Dokter	Total	Rekam_Medis	Total
Rawat_Inap — Rekam_Medis	Rawat_Inap	Total	Rekam_Medis	Parsial
Rawat_Inap — Detail_Rawat_Inap	Rawat_Inap	Total	Detail_Rawat_Inap	Total
Dokter — Detail_Rawat_Inap	Dokter	Total	Detail_Rawat_Inap	Total
Perawat — Detail_Rawat_Inap	Perawat	Total	Detail_Rawat_Inap	Total
Rekam_Medis — Pembayaran	Rekam_Medis	Total	Pembayaran	Total

Pembayaran — Detail_Obat	Pembayaran	Parsial	Detail_Obat	Total
Obat — Detail_Obat	Obat	Total	Detail_Obat	Total

3.5 ERD Final



3.6 Tabel yang dibentuk

Entitas	Attribut	Tipe	keterangan
Pasien	id_pasien	Int (PK)	Identitas unik pasien
	nama_pasien	Varchar	Nama lengkap pasien

	nik	Varchar (unique)	Nomor identitas penduduk
	jenis_kelamin	Enum / char	L/P
	alamat	Text	Alamat lengkap pasien
	no_telp	Varchar	Nomor telepon untuk kontak
	gol_darah	Varchar	Golongan darah pasien

Entitas	Attribut	Tipe	keterangan
Dokter	id_dokter	Int (PK)	
	nama_dokter	Varchar	
	spesialis	Varchar	Bidang spesialisasi dokter
	no_telp	Varchar	Kontak dokter

Entitas	Attribut	Tipe	keterangan
Perawat	id_perawat	Int (PK)	
	nama_perawat	Varchar	
	no_telp	Varchar	

Entitas	Attribut	Tipe	keterangan
Rekam_Medis	id_rekam_medis (PK)	Int (PK)	
	id_pasien (FK)	Int (FK)	Pasien yang diperiksa
	id_dokter (FK)	Int (FK)	Dokter yang menangani
	id_rawat_inap (FK, optional)	Int (FK)	Jika pasien inap
	diagnosa	Varchar(100)	Hasil pemeriksaan
	tindakan	Varchar(100)	Ada tindakan atau resep
	catatan	Varchar(100)	Catatan tambahan
	tgl_rekam	Date	Tanggal pemeriksaan

Entitas	Attribut	Tipe	keterangan
Rawat_Inap	id_rawat_inap (PK)	Int (PK)	
	kamar	Int	Nomor kamar
	kelas	Varchar(3)	Tipe kamar (VIP/1/2/3)
	tanggal_masuk	Date	Masuk rawat
	tanggal_keluar	Date	Keluar rawat
	status_rawat	ENUM('Aktif','Non-Aktif')	Aktif / Non-Aktif

Entitas	Attribut	Tipe	keterangan
Detail_Rawat_Inap	id_detail_rawat (PK)	Int (PK)	

	id_rawat (FK)	Int (FK)	Rawat inap terkait
	id_dokter (FK)	Int (FK)	
	id_perawat (FK)	Int (FK)	
	tgl_mulai	date	Tindakan mulai
	tgl_selesai	date	Tindakan selesai
	catatan	Varchar(100)	Hasil tindakan

Entitas	Attribut	Tipe	keterangan
Pembayaran	id_bayar (PK)	Int (PK)	
	id_rekam (FK)	Int (FK)	Rekam medis terkait
	tgl_bayar	Date	Waktu transaksi
	metode_bayar	Date	Cash / Debit / Transfer
	total_biaya	DECIMAL(15,2)	Total pembayaran
	status_bayar	ENUM('Luunas','Belum')	Lunas / Belum

Entitas	Attribut	Tipe	keterangan
Detail_Obat	id_detail (PK)	Int (PK)	
	id_bayar (FK)	Int (FK)	
	id_obat (FK)	Int (FK)	
	jumlah	Int	Qty
	subtotal	DECIMAL(15,2)	harga_satuan * jumlah

Entitas	Attribut	Tipe	keterangan
Obat	id_obat (PK)	Int (PK)	
	nama_obat	Varchar(100)	Nama obat
	jenis	ENUM('Tablet','Kapsul','Cair')	Golongan obat (Tabel, Kapsul, Cair)
	harga_satuan	DECIMAL(15,2)	Harga per item

BAB IV
IMPLEMENTASI BASIS DATA MENGGUNAKAN MySQL

4.1 Implementasi DDL

Setelah didapati rancangan entitas, atribut, relasi, dan sebagainya, semua kategori diatas atas kita konfersikan kedalam database (SQL) berikut pembuatan database:

```
MariaDB [(none)]> create database klinik_Binasehat;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> use klinik_binasehat;
Database changed
MariaDB [klinik_binasehat]>
```

Setelah database dibuat, kita masuk kedalam database tadi. Karena database ini baru dibuat otomatis data entitas belum ada atau masih kosong, kita inputkan kesulurahan table entitas kedalam database tadi:

```
MariaDB [klinik_binasehat]> CREATE TABLE Pasien (
    ->     id_pasien INT AUTO_INCREMENT PRIMARY KEY,
    ->     nama_pasien VARCHAR(100),
    ->     nik VARCHAR(20),
    ->     jenis_kelamin ENUM('L', 'P'),
    ->     alamat TEXT,
    ->     no_telp VARCHAR(15),
    ->     gol_darah VARCHAR(3)
    -> );
Query OK, 0 rows affected (0.030 sec)
```

```
MariaDB [klinik_binasehat]> CREATE TABLE Dokter (
    ->     id_dokter INT AUTO_INCREMENT PRIMARY KEY,
    ->     nama_dokter VARCHAR(100),
    ->     spesialisasi VARCHAR(100),
    ->     no_telp VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.024 sec)
```

```
MariaDB [klinik_binasehat]> CREATE TABLE Perawat (
    ->     id_perawat INT AUTO_INCREMENT PRIMARY KEY,
    ->     nama VARCHAR(100),
    ->     no_telp VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.012 sec)
```

```
MariaDB [klinik_binasehat]> CREATE TABLE Rekam_Medis (
    ->     id_rekam INT AUTO_INCREMENT PRIMARY KEY,
    ->     id_pasien INT,
    ->     id_dokter INT,
    ->     keluhan TEXT,
    ->     diagnosa TEXT,
    ->     tindakan TEXT,
    ->     tanggal DATE,
    ->     FOREIGN KEY (id_pasien) REFERENCES Pasien(id_pasien),
    ->     FOREIGN KEY (id_dokter) REFERENCES Dokter(id_dokter)
    -> );
Query OK, 0 rows affected (0.029 sec)
```

```
MariaDB [klinik_binasehat]> CREATE TABLE Rawat_Inap (
    ->     id_rawat_inap INT AUTO_INCREMENT PRIMARY KEY,
    ->     kamar VARCHAR(10),
    ->     kelas VARCHAR(20),
    ->     tanggal_masuk DATE,
    ->     tanggal_keluar DATE,
    ->     status_rawat VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.011 sec)
```

```
MariaDB [klinik_binasehat]> CREATE TABLE Detail_Rawat_Inap (
    ->     id_detail_rawat INT AUTO_INCREMENT PRIMARY KEY,
    ->     id_rawat INT,
    ->     id_dokter INT,
    ->     id_perawat INT,
    ->     tgl_mulai DATE,
    ->     tgl_selesai DATE,
    ->     catatan TEXT,
    ->     FOREIGN KEY (id_rawat) REFERENCES Rawat_Inap(id_rawat_inap),
    ->     FOREIGN KEY (id_dokter) REFERENCES Dokter(id_dokter),
    ->     FOREIGN KEY (id_perawat) REFERENCES Perawat(id_perawat)
    -> );
Query OK, 0 rows affected (0.038 sec)
```

```
MariaDB [klinik_binasehat]> CREATE TABLE Pembayaran (
    ->     id_pembayaran INT AUTO_INCREMENT PRIMARY KEY,
    ->     id_pasien INT,
    ->     id_rekam INT,
    ->     total_biaya DECIMAL(12,2),
    ->     status VARCHAR(20),
    ->     FOREIGN KEY (id_pasien) REFERENCES Pasien(id_pasien),
    ->     FOREIGN KEY (id_rekam) REFERENCES Rekam_Medis(id_rekam)
    -> );
Query OK, 0 rows affected (0.032 sec)
```

```
MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> CREATE TABLE Obat (
    ->     id_obat INT AUTO_INCREMENT PRIMARY KEY,
    ->     nama_obat VARCHAR(100),
    ->     jenis VARCHAR(50),
    ->     harga_satuan DECIMAL(12,2)
    -> );
Query OK, 0 rows affected (0.011 sec)
```

```
MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> CREATE TABLE Detail_Pembayaran (
    ->     id_detail INT AUTO_INCREMENT PRIMARY KEY,
    ->     id_pembayaran INT,
    ->     id_obat INT,
    ->     jumlah INT,
    ->     subtotal DECIMAL(12,2),
    ->     FOREIGN KEY (id_pembayaran) REFERENCES Pembayaran(id_pembayaran),
    ->     FOREIGN KEY (id_obat) REFERENCES Obat(id_obat)
    -> );
Query OK, 0 rows affected (0.028 sec)
```

Setelah kita buat table untuk seluruh entitas, kita cek terlebih dahulu apakah sudah sesuai atau tidak entitas di database ini.

```
MariaDB [klinik_binasehat]> show tables;
+---------------------------+
| Tables_in_klinik_binasehat |
+---------------------------+
| detail_pembayaran         |
| detail_rawat_inap         |
| dokter                    |
| obat                      |
| pasien                    |
| pembayaran                |
| perawat                   |
| rawat_inap                |
| rekam_medis               |
+---------------------------+
9 rows in set (0.001 sec)
```

4.2 Implementasi DML

Data Manipulation Language (DML) digunakan untuk memanipulasi data di dalam tabel. Pada tahap ini, dilakukan proses pengisian data awal (*INSERT*), pembaruan data (*UPDATE*), dan penghapusan data (*DELETE*) untuk memastikan sistem dapat mengelola informasi dengan benar.

4.2.1 INSERT (Menambahkan Data) Proses ini memasukkan data master seperti data dokter, obat, dan pasien ke dalam sistem.

```
MariaDB [klinik_binasehat]> INSERT INTO Pasien (nama_pasien, nik, jenis_kelamin, alamat, no_telp, gol_darah) VALUES
    -> ('Ourin', '1234567890123456', 'L', 'Jl. Merpati No. 1, Jakarta', '081234567890', 'A'),
    -> ('Aiko', '2345678901234567', 'P', 'Jl. Kenari No. 2, Bandung', '081345678901', 'B'),
    -> ('Ellen Joe', '3456789012345678', 'P', 'Jl. Kutilang No. 3, Surabaya', '081456789012', 'O'),
    -> ('Gawr Gura', '4567890123456789', 'P', 'Jl. Rajawali No. 4, Semarang', '081567890123', 'AB'),
    -> ('Saba Sameeko', '5678901234567890', 'P', 'Jl. Elang No. 5, Yogyakarta', '081678901234', 'A');
Query OK, 5 rows affected (0.078 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Dokter (nama_dokter, spesialisasi, no_telp) VALUES
    -> ('Shirakaami Fubuki', 'Dokter Umum', '082123456789'),
    -> ('Hutaoo', 'Dokter Bedah', '082234567890'),
    -> ('Exusiai', 'Dokter Anak', '082345678901'),
    -> ('Lucia', 'Dokter Gigi', '082456789012'),
    -> ('Evlyn', 'Dokter Kulit', '082567890123');
Query OK, 5 rows affected (0.005 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Perawat (nama, no_telp) VALUES
    -> ('Furina', '083123456789'),
    -> ('Keqing', '083234567890'),
    -> ('Ourin', '083345678901'),
    -> ('Aiko', '083456789012'),
    -> ('Ellen Joe', '083567890123');
Query OK, 5 rows affected (0.010 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Obat (nama_obat, jenis, harga_satuan) VALUES
    -> ('Paracetamol', 'Analgesik', 2000.00),
    -> ('Amoxicillin', 'Antibiotik', 5000.00),
    -> ('Vitamin C', 'Vitamin', 1500.00),
    -> ('Obat Flu', 'Dekongestan', 2500.00),
    -> ('Salep Luka', 'Topikal', 3000.00);
Query OK, 5 rows affected (0.004 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Rekam_Medis (id_pasien, id_dokter, keluhan, diagnosa, tindakan, tanggal) VALUES
    -> (1, 1, 'Sakit kepala dan demam', 'Flu biasa', 'Pemberian Paracetamol', '2025-12-01'),
    -> (2, 2, 'Batuk dan pilek', 'Infeksi saluran pernapasan', 'Amoxicillin 500mg 3x sehari', '2025-12-02'),
    -> (3, 3, 'Gusi berdarah', 'Radang gusi', 'Pembersihan gigi + Salep Luka', '2025-12-03'),
    -> (4, 4, 'Ruam di kulit', 'Dermatitis', 'Salep Luka dioles 2x sehari', '2025-12-04'),
    -> (5, 5, 'Demam tinggi', 'Infeksi virus', 'Istirahat + Vitamin C', '2025-12-05');
Query OK, 5 rows affected (0.009 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Rawat_Inap (kamar, kelas, tanggal_masuk, tanggal_keluar, status_rawat) VALUES
    -> ('A101', 'VIP', '2025-12-01', '2025-12-03', 'Selesai'),
    -> ('B202', 'Kelas 1', '2025-12-02', '2025-12-06', 'Selesai'),
    -> ('C303', 'Kelas 2', '2025-12-03', NULL, 'Masih Dirawat'),
    -> ('D404', 'VIP', '2025-12-04', NULL, 'Masih Dirawat'),
    -> ('E505', 'Kelas 3', '2025-12-05', '2025-12-07', 'Selesai');
Query OK, 5 rows affected (0.010 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Detail_Rawat_Inap (id_rawat, id_dokter, id_perawat, tgl_mulai, tgl_selesai, cata
tan) VALUES
    -> (1, 1, 1, '2025-12-01', '2025-12-03', 'Pasien stabil, demam turun setelah obat.'),
    -> (2, 2, 2, '2025-12-02', '2025-12-06', 'Batuk berkurang, kondisi membaik.'),
    -> (3, 3, 3, '2025-12-03', NULL, 'Perawatan gusi sedang berjalan.'),
    -> (4, 4, 4, '2025-12-04', NULL, 'Ruam perlahan memudar.'),
    -> (5, 5, 5, '2025-12-05', '2025-12-07', 'Pasien membaik, diperbolehkan pulang.');
Query OK, 5 rows affected (0.009 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Pembayaran (id_pasien, id_rekam, total_biaya, status) VALUES
    -> (1, 1, 70000.00, 'Lunas'),
    -> (2, 2, 125000.00, 'Lunas'),
    -> (3, 3, 24000.00, 'Belum Lunas'),
    -> (4, 4, 55000.00, 'Belum Lunas'),
    -> (5, 5, 30000.00, 'Lunas');
Query OK, 5 rows affected (0.009 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [klinik_binasehat]> INSERT INTO Detail_Pembayaran (id_pembayaran, id_obat, jumlah, subtotal) VALUES
    -> (1, 1, 10, 20000.00),
    -> (1, 3, 25, 50000.00),
    -> (2, 2, 10, 50000.00),
    -> (2, 4, 15, 75000.00),
    -> (3, 5, 4, 24000.00),
    -> (4, 4, 10, 25000.00),
    -> (4, 1, 5, 30000.00),
    -> (5, 3, 20, 30000.00);
Query OK, 8 rows affected (0.010 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

4.2.2 UPDATE (Memperbarui Data) Digunakan ketika terjadi perubahan informasi, misalnya perubahan nama pasien.

```
MariaDB [klinik_binasehat]> select * from pasien;
+-----------+--------------+------------------+---------------+----------------------------+--------------+------------+
| id_pasien | nama_pasien  | nik              | jenis_kelamin | alamat                     | no_telp      | gol_darah  |
+-----------+--------------+------------------+---------------+----------------------------+--------------+------------+
|         1 | Ourin        | 1234567890123456 | L             | Jl. Merpati No. 1, Jakarta | 081234567890 | A          |
|         2 | Aiko         | 2345678901234567 | P             | Jl. Kenari No. 2, Bandung  | 081345678901 | B          |
|         3 | Ellen Joe    | 3456789012345678 | P             | Jl. Kutilang No. 3, Surabaya | 085965911133 | O          |
|         4 | Gawr Gura    | 4567890123456789 | P             | Jl. Rajawali No. 4, Semarang | 081567890123 | AB         |
|         5 | Saba Sameeko | 5678901234567890 | P             | Jl. Elang No. 5, Yogyakarta | 081678901234 | A          |
+-----------+--------------+------------------+---------------+----------------------------+--------------+------------+
5 rows in set (0.025 sec)

MariaDB [klinik_binasehat]> update pasien set nama_pasien = 'Saba Sameko' where id_pasien = 5;
Query OK, 1 row affected (0.017 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [klinik_binasehat]> select * from pasien;
+-----------+--------------+------------------+---------------+----------------------------+--------------+------------+
| id_pasien | nama_pasien  | nik              | jenis_kelamin | alamat                     | no_telp      | gol_darah  |
+-----------+--------------+------------------+---------------+----------------------------+--------------+------------+
|         1 | Ourin        | 1234567890123456 | L             | Jl. Merpati No. 1, Jakarta | 081234567890 | A          |
|         2 | Aiko         | 2345678901234567 | P             | Jl. Kenari No. 2, Bandung  | 081345678901 | B          |
|         3 | Ellen Joe    | 3456789012345678 | P             | Jl. Kutilang No. 3, Surabaya | 085965911133 | O          |
|         4 | Gawr Gura    | 4567890123456789 | P             | Jl. Rajawali No. 4, Semarang | 081567890123 | AB         |
|         5 | Saba Sameko  | 5678901234567890 | P             | Jl. Elang No. 5, Yogyakarta | 081678901234 | A          |
+-----------+--------------+------------------+---------------+----------------------------+--------------+------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]>
```

4.2.3 DELETE (Menghapus Data) Digunakan untuk menghapus data yang tidak diperlukan atau salah input.

```
MariaDB [klinik_binasehat]> select * from obat;
+---------+---------------+-------------+--------------+
| id_obat | nama_obat     | jenis       | harga_satuan |
+---------+---------------+-------------+--------------+
|       1 | Paracetamol   | Analgesik   |      2000.00 |
|       2 | Amoxicillin   | Antibiotik  |      5000.00 |
|       3 | Vitamin C     | Vitamin     |      1500.00 |
|       4 | Obat Flu      | Dekongestan |      2500.00 |
|       5 | Salep Luka    | Topikal     |      3000.00 |
|       6 | Asam Mefenamat | Tablet     |      2500.00 |
+---------+---------------+-------------+--------------+
6 rows in set (0.001 sec)

MariaDB [klinik_binasehat]> delete from obat where id_obat = 6;
ERROR 2013 (HY000): Lost connection to MySQL server during query
MariaDB [klinik_binasehat]> delete from obat where id_obat = 6;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    8
Current database: klinik_binasehat

Query OK, 1 row affected (0.012 sec)

MariaDB [klinik_binasehat]> select * from obat;
+---------+-------------+-------------+--------------+
| id_obat | nama_obat   | jenis       | harga_satuan |
+---------+-------------+-------------+--------------+
|       1 | Paracetamol | Analgesik   |      2000.00 |
|       2 | Amoxicillin | Antibiotik  |      5000.00 |
|       3 | Vitamin C   | Vitamin     |      1500.00 |
|       4 | Obat Flu    | Dekongestan |      2500.00 |
|       5 | Salep Luka  | Topikal     |      3000.00 |
+---------+-------------+-------------+--------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]>
```

4.3 Query JOIN

Query *JOIN* digunakan untuk menggabungkan dua tabel atau lebih berdasarkan kolom kunci yang saling berelasi (*Foreign Key*). Berikut adalah implementasi *INNER JOIN* untuk menampilkan data Rekam Medis lengkap dengan nama Pasien, sehingga informasi mudah dibaca.

```
MariaDB [klinik_binasehat]> SELECT
    ->     p.id_pasien,
    ->     p.nama_pasien,
    ->     r.keluhan,
    ->     r.diagnosa,
    ->     r.tanggal
    -> FROM pasien p
    -> INNER JOIN rekam_medis r
    ->     ON p.id_pasien = r.id_pasien;
+------------+--------------+------------------------+----------------------------+------------+
| id_pasien  | nama_pasien  | keluhan                | diagnosa                   | tanggal    |
+------------+--------------+------------------------+----------------------------+------------+
|          1 | Ourin        | Sakit kepala dan demam | Flu biasa                  | 2025-12-01 |
|          2 | Aiko         | Batuk dan pilek        | Infeksi saluran pernapasan | 2025-12-02 |
|          3 | Ellen Joe    | Gusi berdarah          | Radang gusi                | 2025-12-03 |
|          4 | Gawr Gura    | Ruam di kulit          | Dermatitis                 | 2025-12-04 |
|          5 | Saba Sameeko | Demam tinggi           | Infeksi virus              | 2025-12-05 |
+------------+--------------+------------------------+----------------------------+------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]>
```

4.4 Subquery

Dalam pengelolaan basis data yang kompleks, seringkali diperlukan pengambilan informasi spesifik yang nilainya bergantung pada hasil pemrosesan data lain secara dinamis. Untuk menangani kebutuhan tersebut, pada bagian ini diterapkan teknik *Subquery* atau *nested query*.

Secara spesifik, *subquery* dapat digunakan untuk menyeleksi data pasien yang memiliki total riwayat transaksi pembayaran di atas nilai rata-rata pendapatan klinik. Pendekatan ini memungkinkan manajemen untuk mengidentifikasi segmen pasien

dengan kontribusi finansial tertinggi tanpa perlu melakukan perhitungan manual yang rentan kesalahan.

```
MariaDB [klinik_binasehat]> SELECT nama_pasien, no_telp FROM Pasien WHERE id_pasien IN ( SELECT id_pasien FROM rekam_medis WHERE id_rekam in ( s
elect id_rekam from pembayaran where total_biaya > (SELECT AVG(total_biaya) FROM Pembayaran)));
+-------------+--------------+
| nama_pasien | no_telp      |
+-------------+--------------+
| Ourin       | 081234567890 |
| Aiko        | 081345678901 |
+-------------+--------------+
2 rows in set (0.006 sec)

MariaDB [klinik_binasehat]>
```

4.5 Penggunaan Fungsi Agregat

Guna mendukung proses pengambilan keputusan manajerial yang berbasis data (*data-driven decision making*), sistem dilengkapi dengan kemampuan analisis statistik deskriptif melalui pemanfaatan fungsi agregat pada basis data. Fungsi-fungsi agregat seperti COUNT, SUM, AVG, MAX, dan MIN diimplementasikan untuk menghasilkan ringkasan data operasional, meliputi perhitungan jumlah pasien, rata-rata total pembayaran pasien, akumulasi pendapatan serta laba kotor klinik, serta analisis harga obat tertinggi dan terendah yang beredar dalam sistem.

```
MariaDB [klinik_binasehat]> select max(harga_satuan) as Harga_Obat_Termahal from obat;
+---------------------+
| Harga_Obat_Termahal |
+---------------------+
|             5000.00 |
+---------------------+
1 row in set (0.001 sec)
```

```
MariaDB [klinik_binasehat]> select min(harga_satuan) as Harga_Obat_Termurah from obat;
+---------------------+
| Harga_Obat_Termurah |
+---------------------+
|             1500.00 |
+---------------------+
1 row in set (0.001 sec)
```

```
MariaDB [klinik_binasehat]> select AVG(total_biaya) as Rata_Pembayaran from Pembayaran;
+-----------------+
| Rata_Pembayaran |
+-----------------+
|    60800.000000 |
+-----------------+
1 row in set (0.000 sec)
```

```
MariaDB [klinik_binasehat]> select SUM(total_biaya) as laba_kotor from Pembayaran;
+------------+
| laba_kotor |
+------------+
|  304000.00 |
+------------+
1 row in set (0.001 sec)
```

```
MariaDB [klinik_binasehat]> select count(id_pasien) as jumlah_pasein from pasien;
+---------------+
| jumlah_pasein |
+---------------+
|             5 |
+---------------+
1 row in set (0.001 sec)
```

4.6 View

Untuk meningkatkan efisiensi akses data serta menyederhanakan struktur *query* yang kompleks bagi pengguna akhir, diimplementasikan fitur *View*. Sebuah objek virtual bernama Laporan_Harian dibentuk untuk mengenkapsulasi logika *JOIN* antara tabel Rekam Medis, Pasien, dan Dokter. Dengan pendekatan ini, integritas struktur query terjaga dan pengguna dapat mengakses laporan lengkap hanya dengan memanggil nama *View* tersebut tanpa perlu menulis ulang sintaks SQL yang rumit.

```
MariaDB [klinik_binasehat]> create view DataLaporan as select rm.id_rekam,rm.tanggal, p.nama_pasien,d.nama_dokter,rm.diagnosa from Rekam_Medis r
m join Pasien p ON rm.id_pasien = p.id_pasien JOIN Dokter d ON rm.id_dokter = d.id_dokter;
Query OK, 0 rows affected (0.009 sec)

MariaDB [klinik_binasehat]> select * from DataLaporan;
+-----------+------------+-------------+-----------------+----------------------------+
| id_rekam  | tanggal    | nama_pasien | nama_dokter     | diagnosa                   |
+-----------+------------+-------------+-----------------+----------------------------+
|         1 | 2025-12-01 | Ourin       | Shirakaami Fubuki | Flu biasa                |
|         2 | 2025-12-02 | Aiko        | Hutaoo          | Infeksi saluran pernapasan |
|         3 | 2025-12-03 | Ellen Joe   | Exusiai         | Radang gusi                |
|         4 | 2025-12-04 | Gawr Gura   | Lucia           | Dermatitis                 |
|         5 | 2025-12-05 | Saba Sameko | Evlyn           | Infeksi virus              |
+-----------+------------+-------------+-----------------+----------------------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]>
```

4.7 Trigger

Untuk memungkinkan basis data menjalankan suatu operasi secara otomatis berdasarkan peristiwa tertentu, sistem memanfaatkan trigger. Trigger merupakan mekanisme yang akan dieksekusi secara otomatis ketika terjadi suatu *event* pada tabel, seperti INSERT, UPDATE, atau DELETE, sesuai dengan fungsi yang telah didefinisikan. Penggunaan trigger bertujuan untuk mendukung berbagai kebutuhan sistem, antara lain pencatatan log perubahan data, perhitungan otomatis, serta menjaga konsistensi data tanpa memerlukan intervensi langsung dari aplikasi.

4.7.1 Trigger Pembaruan Total Pembayaran Pasien

Salah satu implementasi trigger dalam sistem ini adalah trigger untuk memperbarui total pembayaran pasien secara otomatis ketika terjadi penambahan data obat pada tabel detail pembelian obat. Trigger ini dijalankan setelah proses INSERT, sehingga setiap penambahan obat baru akan langsung memengaruhi nilai total pembayaran pada tabel pembayaran yang bersangkutan.

```
MariaDB [klinik_binasehat]> DELIMITER $$
MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> CREATE TRIGGER UPD_TOTAL_PEMBAYARAN
    -> AFTER INSERT ON Detail_Pembayaran
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Pembayaran
    ->     SET total_biaya = total_biaya + NEW.subtotal
    ->     WHERE id_pembayaran = NEW.id_pembayaran;
    -> END$$
Query OK, 0 rows affected (0.018 sec)

MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> DELIMITER ;
```

```
MariaDB [klinik_binasehat]> select * from pembayaran;
+---------------+-----------+----------+-------------+-------------+
| id_pembayaran | id_pasien | id_rekam | total_biaya | status      |
+---------------+-----------+----------+-------------+-------------+
|             1 |         1 |        1 |    70000.00 | Lunas       |
|             2 |         2 |        2 |   125000.00 | Lunas       |
|             3 |         3 |        3 |    24000.00 | Belum Lunas |
|             4 |         4 |        4 |    55000.00 | Belum Lunas |
|             5 |         5 |        5 |    30000.00 | Lunas       |
+---------------+-----------+----------+-------------+-------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]> select * from detail_pembayaran;
+-----------+---------------+---------+--------+----------+
| id_detail | id_pembayaran | id_obat | jumlah | subtotal |
+-----------+---------------+---------+--------+----------+
|         1 |             1 |       1 |     10 | 20000.00 |
|         2 |             1 |       3 |     25 | 50000.00 |
|         3 |             2 |       2 |     10 | 50000.00 |
|         4 |             2 |       4 |     15 | 75000.00 |
|         5 |             3 |       5 |      4 | 24000.00 |
|         6 |             4 |       4 |     10 | 25000.00 |
|         7 |             4 |       1 |      5 | 30000.00 |
|         8 |             5 |       3 |     20 | 30000.00 |
+-----------+---------------+---------+--------+----------+
8 rows in set (0.007 sec)

MariaDB [klinik_binasehat]>
```

Setelah dilakukan proses insert data pada tabel detail pembelian, sistem secara otomatis melakukan perhitungan ulang total pembayaran. Hal ini dapat dilihat pada data pembayaran, dengan ID pembayaran 5, di mana nilai total pembayaran berubah secara otomatis setelah data baru ditambahkan pada tabel detail pembelian.

```
MariaDB [klinik_binasehat]> INSERT INTO Detail_Pembayaran (id_pembayaran, id_obat, jumlah, subtotal) VALUES
    -> (5, 4, 10, 25000.00);
Query OK, 1 row affected (0.006 sec)

MariaDB [klinik_binasehat]> select * from detail_pembayaran;
+-----------+---------------+---------+--------+----------+
| id_detail | id_pembayaran | id_obat | jumlah | subtotal |
+-----------+---------------+---------+--------+----------+
|         1 |             1 |       1 |     10 | 20000.00 |
|         2 |             1 |       3 |     25 | 50000.00 |
|         3 |             2 |       2 |     10 | 50000.00 |
|         4 |             2 |       4 |     15 | 75000.00 |
|         5 |             3 |       5 |      4 | 24000.00 |
|         6 |             4 |       4 |     10 | 25000.00 |
|         7 |             4 |       1 |      5 | 30000.00 |
|         8 |             5 |       3 |     20 | 30000.00 |
|         9 |             5 |       4 |     10 | 25000.00 |
+-----------+---------------+---------+--------+----------+
9 rows in set (0.000 sec)

MariaDB [klinik_binasehat]> select * from pembayaran;
+---------------+-----------+----------+-------------+-------------+
| id_pembayaran | id_pasien | id_rekam | total_biaya | status      |
+---------------+-----------+----------+-------------+-------------+
|             1 |         1 |        1 |    70000.00 | Lunas       |
|             2 |         2 |        2 |   125000.00 | Lunas       |
|             3 |         3 |        3 |    24000.00 | Belum Lunas |
|             4 |         4 |        4 |    55000.00 | Belum Lunas |
|             5 |         5 |        5 |    55000.00 | Lunas       |
+---------------+-----------+----------+-------------+-------------+
5 rows in set (0.000 sec)

MariaDB [klinik_binasehat]>
```

4.7.2 Trigger Pencatatan Log Perubahan Data Pasien

Selain perhitungan otomatis, trigger juga digunakan untuk mencatat riwayat perubahan data pasien. Untuk keperluan ini, sistem menyediakan sebuah tabel khusus yang berfungsi sebagai tabel log pembaruan data pasien. Pada kondisi awal, tabel log tersebut masih kosong.

Selanjutnya, dibuat sebuah trigger yang akan dijalankan ketika terjadi proses UPDATE pada tabel pasien.

```
MariaDB [klinik_binasehat]> CREATE TABLE log_perubahan_pasien (
    ->     id_log INT AUTO_INCREMENT PRIMARY KEY,
    ->     id_pasien INT,
    ->     nama_lama VARCHAR(100),
    ->     nama_baru VARCHAR(100),
    ->     alamat_lama VARCHAR(255),
    ->     alamat_baru VARCHAR(255),
    ->     telp_lama VARCHAR(20),
    ->     telp_baru VARCHAR(20),
    ->     waktu_perubahan TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.027 sec)

MariaDB [klinik_binasehat]> select * from pasien;
+-----------+--------------+------------------+--------------+-----------------------------+-------------+-----------+
| id_pasien | nama_pasien  | nik              | jenis_kelamin | alamat                      | no_telp     | gol_darah |
+-----------+--------------+------------------+--------------+-----------------------------+-------------+-----------+
|         1 | Ourin        | 1234567890123456 | L            | Jl. Merpati No. 1, Jakarta  | 081234567890 | A         |
|         2 | Aiko         | 2345678901234567 | P            | Jl. Kenari No. 2, Bandung   | 081345678901 | B         |
|         3 | Ellen Joe    | 3456789012345678 | P            | Jl. Kutilang No. 3, Surabaya | 081456789012 | O         |
|         4 | Gawr Gura    | 4567890123456789 | P            | Jl. Rajawali No. 4, Semarang | 081567890123 | AB        |
|         5 | Saba Sameeko | 5678901234567890 | P            | Jl. Elang No. 5, Yogyakarta | 081678901234 | A         |
+-----------+--------------+------------------+--------------+-----------------------------+-------------+-----------+
5 rows in set (0.008 sec)

MariaDB [klinik_binasehat]> select * from log_perubahan_pasien;
Empty set (0.002 sec)
```

```
MariaDB [klinik_binasehat]> DELIMITER $$
MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> CREATE TRIGGER log_update_pasien
    -> AFTER UPDATE ON pasien
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO log_perubahan_pasien
    ->     (id_pasien, nama_lama, nama_baru, alamat_lama, alamat_baru, telp_lama, telp_baru, waktu_perubahan)
    ->     VALUES
    ->     (OLD.id_pasien, OLD.nama_pasien, NEW.nama_pasien, OLD.alamat, NEW.alamat, OLD.no_telp, NEW.no_telp, NOW());
    -> END $$
Query OK, 0 rows affected (0.018 sec)

MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> DELIMITER ;
```

Sebagai contoh, dilakukan pembaruan data nomor telepon pasien dengan ID pasien 3 atas nama Ellen Joe. Setelah proses pembaruan dilakukan, data nomor telepon pada tabel pasien berubah sesuai dengan nilai baru, dan perubahan tersebut secara otomatis tercatat pada tabel log pembaruan pasien.

```
MariaDB [klinik_binasehat]> update pasien set no_telp = '085965911133' where id_pasien = 3;
Query OK, 1 row affected (0.012 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [klinik_binasehat]> select * from pasien;
+-----------+--------------+------------------+---------------+-----------------------------+--------------+------------+
| id_pasien | nama_pasien  | nik              | jenis_kelamin | alamat                      | no_telp      | gol_darah  |
+-----------+--------------+------------------+---------------+-----------------------------+--------------+------------+
|         1 | Ourin        | 12345678901234567 | L             | Jl. Merpati No. 1, Jakarta  | 081234567890 | A          |
|         2 | Aiko         | 2345678901234567 | P             | Jl. Kenari No. 2, Bandung   | 081345678901 | B          |
|         3 | Ellen Joe    | 3456789012345678 | P             | Jl. Kutilang No. 3, Surabaya | 085965911133 | O          |
|         4 | Gawr Gura    | 4567890123456789 | P             | Jl. Rajawali No. 4, Semarang | 081567890123 | AB         |
|         5 | Saba Sameeko | 5678901234567890 | P             | Jl. Elang No. 5, Yogyakarta | 081678901234 | A          |
+-----------+--------------+------------------+---------------+-----------------------------+--------------+------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]> select * from log_perubahan_pasien;
+--------+-----------+------------+------------+-----------------------------+-----------------------------+--------------+--------------+---------------------+
| id_log | id_pasien | nama_lama  | nama_baru  | alamat_lama                 | alamat_baru                 | telp_lama    | telp_baru    | waktu_perubahan     |
+--------+-----------+------------+------------+-----------------------------+-----------------------------+--------------+--------------+---------------------+
|      1 |         3 | Ellen Joe  | Ellen Joe  | Jl. Kutilang No. 3, Surabaya | Jl. Kutilang No. 3, Surabaya | 081456789012 | 085965911133 | 2025-12-08 11:56:33 |
+--------+-----------+------------+------------+-----------------------------+-----------------------------+--------------+--------------+---------------------+
1 row in set (0.001 sec)

MariaDB [klinik_binasehat]>
```

Dengan adanya mekanisme ini, sistem mampu melacak setiap perubahan data secara historis, sehingga meningkatkan transparansi, akuntabilitas, serta keamanan pengelolaan data dalam basis data.

4.8 Stored Procedure

Untuk mempermudah pelaksanaan suatu proses dalam basis data, sistem memanfaatkan stored procedure. Stored procedure merupakan sekumpulan perintah SQL yang disimpan di dalam basis data dan dapat dipanggil kembali hanya dengan menyebutkan nama prosedur beserta parameter yang sesuai. Pendekatan ini bertujuan untuk meningkatkan efisiensi, keterbacaan, serta konsistensi dalam pengelolaan data.

Salah satu implementasi stored procedure pada sistem ini adalah prosedur untuk penambahan data obat. Dengan adanya stored procedure, proses penambahan obat tidak perlu lagi dilakukan melalui perintah INSERT INTO secara berulang. Seluruh logika penambahan data dirangkum ke dalam satu prosedur, sehingga proses menjadi lebih ringkas dan terstruktur.

```
MariaDB [klinik_binasehat]> DELIMITER $$
MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> CREATE PROCEDURE tambah_obat (
    ->     IN p_nama_obat VARCHAR(100),
    ->     IN p_jenis VARCHAR(50),
    ->     IN p_harga DECIMAL(12,2)
    -> )
    -> BEGIN
    ->     INSERT INTO Obat (nama_obat, jenis, harga_satuan)
    ->     VALUES (p_nama_obat, p_jenis, p_harga);
    -> END $$
Query OK, 0 rows affected (0.034 sec)

MariaDB [klinik_binasehat]>
MariaDB [klinik_binasehat]> DELIMITER ;
```

Dengan demikian, ketika pengguna ingin menambahkan data obat baru, sistem cukup memanggil nama stored procedure yang telah dibuat dengan menyertakan parameter yang diperlukan. Cara ini tidak hanya menyederhanakan proses input data, tetapi juga mengurangi potensi kesalahan penulisan perintah SQL serta meningkatkan kemudahan pemeliharaan sistem basis data.

```
MariaDB [klinik_binasehat]> select * from obat;
+---------+-------------+-------------+--------------+
| id_obat | nama_obat   | jenis       | harga_satuan |
+---------+-------------+-------------+--------------+
|       1 | Paracetamol | Analgesik   |      2000.00 |
|       2 | Amoxicillin | Antibiotik  |      5000.00 |
|       3 | Vitamin C   | Vitamin     |      1500.00 |
|       4 | Obat Flu    | Dekongestan |      2500.00 |
|       5 | Salep Luka  | Topikal     |      3000.00 |
+---------+-------------+-------------+--------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]> CALL tambah_obat('Asam Mefenamat', 'Tablet', 2500);
Query OK, 1 row affected (0.008 sec)

MariaDB [klinik_binasehat]> select * from obat;
+---------+---------------+-------------+--------------+
| id_obat | nama_obat     | jenis       | harga_satuan |
+---------+---------------+-------------+--------------+
|       1 | Paracetamol   | Analgesik   |      2000.00 |
|       2 | Amoxicillin   | Antibiotik  |      5000.00 |
|       3 | Vitamin C     | Vitamin     |      1500.00 |
|       4 | Obat Flu      | Dekongestan |      2500.00 |
|       5 | Salep Luka    | Topikal     |      3000.00 |
|       6 | Asam Mefenamat | Tablet      |      2500.00 |
+---------+---------------+-------------+--------------+
6 rows in set (0.001 sec)

MariaDB [klinik_binasehat]>
```

4.9 Group by dan Having

Pengelompokan data untuk kebutuhan pelaporan analitis dilakukan menggunakan klausa *Group By* dan *Having*. Metode ini diterapkan untuk menganalisis beban kerja tenaga medis dengan mengelompokkan data rekam medis berdasarkan dokter yang menanganinya. Selanjutnya, filter *Having* diaplikasikan untuk menyaring dan menampilkan hanya dokter yang memiliki intensitas pelayanan tinggi (menangani lebih dari sejumlah pasien tertentu), sehingga manajemen dapat melakukan evaluasi kinerja SDM secara efektif.

```
MariaDB [klinik_binasehat]> select d.nama_dokter,
    -> count(rm.id_pasien) as jumlah_pasien
    -> from dokter d
    -> join rekam_medis rm on d.id_dokter = rm.id_dokter
    -> GROUP BY d.nama_dokter
    -> HAVING jumlah_pasien >= 1;
+-------------------+---------------+
| nama_dokter       | jumlah_pasien |
+-------------------+---------------+
| Evlyn             |             1 |
| Exusiai           |             1 |
| Hutaoo            |             1 |
| Lucia             |             1 |
| Shirakaami Fubuki |             1 |
+-------------------+---------------+
5 rows in set (0.016 sec)

MariaDB [klinik_binasehat]>
```

4.10 Transaction Control (Commit/Rollback)

Dalam pengelolaan transaksi keuangan pada sistem, diperlukan mekanisme yang mampu memastikan setiap proses berjalan secara konsisten dan terkontrol. Untuk memenuhi kebutuhan tersebut, sistem menerapkan Transaction Control Language (TCL) sebagai pengendali jalannya transaksi pembayaran. Mengingat proses pembayaran melibatkan beberapa operasi yang saling berkaitan, seperti pembaruan data pembayaran dan stok secara bersamaan, maka digunakan prinsip atomisitas transaksi dengan perintah START TRANSACTION, COMMIT, dan ROLLBACK. Mekanisme ini memungkinkan sistem membatalkan seluruh perubahan data apabila terjadi kegagalan di tengah proses transaksi, sehingga kondisi data tetap terjaga dan tidak menimbulkan ketidaksesuaian pada data keuangan klinik.

```
MariaDB [klinik_binasehat]> START TRANSACTION;
Query OK, 0 rows affected (0.001 sec)

MariaDB [klinik_binasehat]> update pembayaran set status = 'Lunas' where id_pembayaran = 3;
Query OK, 1 row affected (0.015 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [klinik_binasehat]> select * from pembayaran;
+---------------+-----------+----------+-------------+-------------+
| id_pembayaran | id_pasien | id_rekam | total_biaya | status      |
+---------------+-----------+----------+-------------+-------------+
|             1 |         1 |        1 |    70000.00 | Lunas       |
|             2 |         2 |        2 |   125000.00 | Lunas       |
|             3 |         3 |        3 |    24000.00 | Lunas       |
|             4 |         4 |        4 |    55000.00 | Belum Lunas |
|             5 |         5 |        5 |    55000.00 | Lunas       |
+---------------+-----------+----------+-------------+-------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]> ROLLBACK;
Query OK, 0 rows affected (0.006 sec)

MariaDB [klinik_binasehat]> select * from pembayaran;
+---------------+-----------+----------+-------------+-------------+
| id_pembayaran | id_pasien | id_rekam | total_biaya | status      |
+---------------+-----------+----------+-------------+-------------+
|             1 |         1 |        1 |    70000.00 | Lunas       |
|             2 |         2 |        2 |   125000.00 | Lunas       |
|             3 |         3 |        3 |    24000.00 | Belum Lunas |
|             4 |         4 |        4 |    55000.00 | Belum Lunas |
|             5 |         5 |        5 |    55000.00 | Lunas       |
+---------------+-----------+----------+-------------+-------------+
5 rows in set (0.002 sec)

MariaDB [klinik_binasehat]>
```

```
MariaDB [klinik_binasehat]> START TRANSACTION;
Query OK, 0 rows affected (0.001 sec)

MariaDB [klinik_binasehat]> update pembayaran set status = 'Lunas' where id_pembayaran = 3;
Query OK, 1 row affected (0.009 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [klinik_binasehat]> select * from pembayaran;
+---------------+-----------+----------+-------------+-------------+
| id_pembayaran | id_pasien | id_rekam | total_biaya | status      |
+---------------+-----------+----------+-------------+-------------+
|             1 |         1 |        1 |    70000.00 | Lunas       |
|             2 |         2 |        2 |   125000.00 | Lunas       |
|             3 |         3 |        3 |    24000.00 | Lunas       |
|             4 |         4 |        4 |    55000.00 | Belum Lunas |
|             5 |         5 |        5 |    55000.00 | Lunas       |
+---------------+-----------+----------+-------------+-------------+
5 rows in set (0.001 sec)

MariaDB [klinik_binasehat]> commit:
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax
to use near ':' at line 1
MariaDB [klinik_binasehat]> Commit;
Query OK, 0 rows affected (0.007 sec)

MariaDB [klinik_binasehat]> select * from pembayaran;
+---------------+-----------+----------+-------------+-------------+
| id_pembayaran | id_pasien | id_rekam | total_biaya | status      |
+---------------+-----------+----------+-------------+-------------+
|             1 |         1 |        1 |    70000.00 | Lunas       |
|             2 |         2 |        2 |   125000.00 | Lunas       |
|             3 |         3 |        3 |    24000.00 | Lunas       |
|             4 |         4 |        4 |    55000.00 | Belum Lunas |
|             5 |         5 |        5 |    55000.00 | Lunas       |
+---------------+-----------+----------+-------------+-------------+
5 rows in set (0.001 sec)
```

4.11 Hak Akses + Screenshot + Screenshoot

Pengendalian hak akses pengguna merupakan bagian penting dalam menjaga keamanan sistem informasi medis. Pada sistem ini, pengelolaan hak akses dilakukan menggunakan Data Control Language (DCL) untuk mengatur kewenangan setiap pengguna sesuai dengan perannya. Penerapan konsep Least Privilege digunakan sebagai strategi utama, di mana setiap pengguna hanya diberikan hak akses minimum yang diperlukan untuk menjalankan tugasnya.

Melalui pendekatan ini, sistem membedakan hak akses berdasarkan peran pengguna, seperti dokter, perawat, maupun administrator. Sebagai contoh, pengguna dengan peran Dokter hanya diberikan hak untuk membaca (SELECT) dan menambahkan (INSERT) data rekam medis, tanpa memiliki izin untuk mengubah struktur tabel, menghapus data, atau mengakses tabel data master lainnya. Dengan pembatasan tersebut, kerahasiaan serta integritas data pasien dapat tetap terjaga dari akses yang tidak sah.

5.1 Pembuatan Akun Baru (Dr.Lucia)

```
Banu Najieh alFadhil@OURIN c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create user 'Dr_Lucia'@'localhost' IDENTIFIED BY '12345';
Query OK, 0 rows affected (0.017 sec)
```

```
Banu Najieh alFadhil@OURIN c:\xampp
# mysql -u Dr_Lucia -p
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 18
Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.001 sec)

MariaDB [(none)]>
```

5.2 Pengecekan Hak Akses Awal Pengguna

```
MariaDB [klinik_binasehat]> SHOW GRANTS FOR 'Dr_Lucia'@'localhost';
+-----------------------------------------------------------------------------------------------------------------+
| Grants for Dr_Lucia@localhost                                                                                   |
+-----------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `Dr_Lucia`@`localhost` IDENTIFIED BY PASSWORD '*00A51F3F48415C7D4E8908980D443C29C69B60C9' |
+-----------------------------------------------------------------------------------------------------------------+
1 row in set (0.004 sec)

MariaDB [klinik_binasehat]>
```

5.3 Pemberian Hak Akses Rekam Medis Untuk Dr Lucia

```
MariaDB [klinik_binasehat]> GRANT SELECT, INSERT ON Rekam_Medis TO 'Dr_Lucia'@'localhost';
Query OK, 0 rows affected (0.010 sec)

MariaDB [klinik_binasehat]> SHOW GRANTS FOR 'Dr_Lucia'@'localhost';
+-----------------------------------------------------------------------------------------------------------------+
| Grants for Dr_Lucia@localhost                                                                                   |
+-----------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `Dr_Lucia`@`localhost` IDENTIFIED BY PASSWORD '*00A51F3F48415C7D4E8908980D443C29C69B60C9' |
| GRANT SELECT, INSERT ON `klinik_binasehat`.`rekam_medis` TO `Dr_Lucia`@`localhost`                              |
+-----------------------------------------------------------------------------------------------------------------+
2 rows in set (0.000 sec)

MariaDB [klinik_binasehat]>
```

5.4 Pengecekan Privilage Pada Dr Lucia

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| klinik_binasehat   |
+--------------------+
2 rows in set (0.003 sec)

MariaDB [(none)]> use klinik_binasehat;
Database changed
MariaDB [klinik_binasehat]> show tables;
+----------------------------+
| Tables_in_klinik_binasehat |
+----------------------------+
| rekam_medis                |
+----------------------------+
1 row in set (0.006 sec)

MariaDB [klinik_binasehat]>
```

BAB V
PENGUJIAN DAN VALIDASI SISTEM

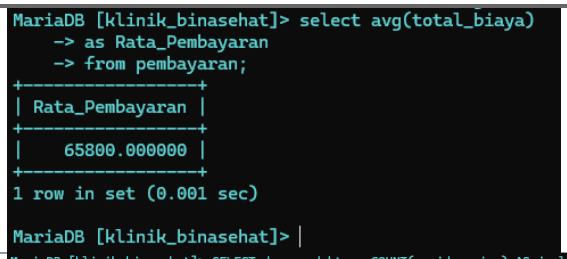
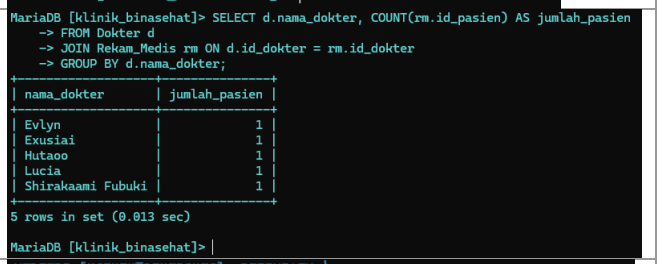
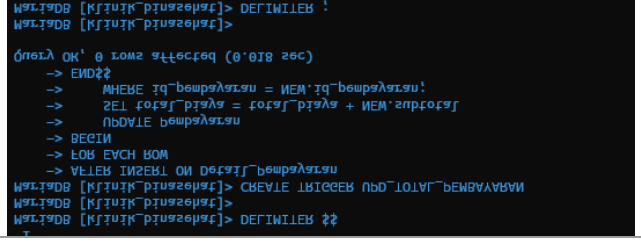
5.1 Tujuan Pengujian

Tahapan pengujian dan validasi sistem bertujuan untuk memverifikasi fungsionalitas dan integritas arsitektur basis data yang telah diimplementasikan. Secara spesifik, pengujian ini difokuskan untuk memastikan bahwa:

1. Seluruh aturan batasan (*constraints*) seperti Primary Key, Foreign Key, berfungsi dalam mencegah anomali data.
2. Fitur otomatisasi prosedural (*Triggers* dan *Stored Procedures*) tereksekusi secara akurat sesuai dengan alur bisnis klinik.
3. Mekanisme kontrol akses pengguna (*User Privileges*) berjalan efektif dalam melindungi kerahasiaan data sensitif.

5.2 Daftar Test Case

Pengujian sistem dilakukan dengan cara menguji fungsi-fungsi yang tersedia melalui pemberian input tertentu dan mengamati output yang dihasilkan. Pendekatan ini digunakan untuk memastikan bahwa sistem bekerja sesuai dengan spesifikasi yang diharapkan. Matriks skenario pengujian berikut disusun untuk menggambarkan pengujian terhadap beberapa proses utama dalam sistem.

No	Problem	Result
1	Manajemen kesulitan mengetahui total pendapatan dari seluruh transaksi pembayaran pasien.	
2	Sistem tidak dapat menampilkan jumlah pasien yang ditangani oleh setiap dokter.	
3	Total pembayaran pasien tidak otomatis berubah ketika terdapat penambahan data obat pada detail pembelian, sehingga berpotensi menimbulkan kesalahan perhitungan biaya.	

5.3 Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang dilakukan, sistem basis data klinik mampu mengatasi berbagai permasalahan operasional yang ada. Sistem dapat menampilkan total pendapatan dari seluruh transaksi pembayaran pasien secara cepat dan akurat melalui pemanfaatan fungsi agregat, sehingga manajemen tidak perlu melakukan perhitungan manual. Selain itu, jumlah pasien yang ditangani oleh setiap dokter dapat diketahui dengan jelas menggunakan relasi antar tabel dan proses pengelompokan data. Penerapan trigger juga terbukti efektif dalam memperbarui total pembayaran pasien secara otomatis ketika terjadi penambahan data obat, sehingga risiko kesalahan perhitungan biaya dapat diminimalkan dan konsistensi data tetap terjaga.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisis, perancangan, dan implementasi yang telah dilakukan, dapat disimpulkan beberapa poin utama sebagai berikut:

a) Basis data relasional yang dirancang mampu mengurangi duplikasi data antara unit Poli dan Farmasi, sehingga informasi tersimpan secara terpusat dan lebih konsisten.
b) Penggunaan fitur DBMS seperti Trigger untuk pengelolaan stok dan View untuk pelaporan rekam medis membantu menyederhanakan proses operasional serta mengurangi risiko kesalahan akibat input manual
c) Penerapan manajemen hak akses berbasis peran (Role-Based Access Control) memberikan perlindungan yang cukup baik terhadap data medis pasien dari akses yang tidak berwenang.

6.2 Saran

Untuk menjaga agar sistem dapat terus dikembangkan dan digunakan dalam jangka panjang, beberapa pengembangan lanjutan (future work) yang dapat dilakukan antara lain:

a) Perlu dikembangkan antarmuka pengguna (GUI) berbasis web atau mobile yang mudah digunakan, agar pengguna non-teknis dapat berinteraksi dengan sistem basis data secara lebih nyaman
b) Diperlukan penerapan mekanisme pencadangan data otomatis secara berkala serta prosedur pemulihan sistem (Disaster Recovery) untuk mengurangi risiko kehilangan data akibat gangguan atau kerusakan perangkat keras.