

Laporan Akhir Project Basis Data



Dosen Pengampu : Nasrul, S.Pd.I, S.Kom, M.Kom

Asisten Dosen : Yuliadi, Amd

Data Kelompok

Nama Lengkap	NIM
Diah Ayu Puspasari	0110223052
Nur fadillah	0110223075
Fatimah Labibah	0110223060
Muhammad Fajar	0110223078
Muhammad Zen Alby	0110223077

Teknik Informatika 2302

MERANCANG DATABASE RUMAH SAKIT

**STT - TERPADU NURUL FIKRI
TAHUN AJARAN 2023
SEMESTER GENAP**

DAFTAR ISI

I. Latar Belakang	3
II. Deskripsi Tugas Akhir	3
III. Desain Database dengan mysql workbench	4
IV. Struktur Database	5
1) Tabel Divisi	5
2) Tabel Pegawai	5
3) Tabel Pasien.....	6
4) Tabel Dokter	6
5) Tabel Pendaftaran	7
6) Tabel Periksa	7
7) Tabel Ruangan.....	7
8) Tabel Rawat Inap.....	8
9) Tabel Obat	8
10) Tabel Pembayaran.....	9
V. Relasi Table	10
VI. Pembahasan Query.....	13
a. CRUD.....	13
b. INNER JOIN	14
c. OUTER JOIN.....	15
d. INDEX	16
e. VIEW	17
f. TRANSACTION COMMIT	19
g. ROLLBACK	20
h. PROCEDURE	22
i. TRIGGER.....	23
j. Data Controlling Language (DCL)	25
k. BACKUP.....	26
l. RESTORE DATABASE.....	27

I. Latar Belakang

Proyek ini bertujuan untuk mengelola dan mengoptimalkan sistem informasi di sebuah rumah sakit yang mencakup berbagai aspek penting seperti manajemen pegawai, pasien, pemeriksaan, obat-obatan, dan pembayaran. Rumah sakit adalah sebuah entitas yang kompleks dengan kebutuhan administratif dan operasional yang tinggi, dimana pengelolaan data yang efisien sangatlah krusial untuk memastikan layanan kesehatan yang berkualitas bagi pasien.

- 1) **Manajemen Pegawai:** Data pegawai meliputi informasi tentang nama, NIP, divisi tempat mereka bekerja, nomor telepon, email, tanggal lahir, jenis kelamin, dan alamat. Manajemen yang efektif terhadap data pegawai memungkinkan pengelolaan jadwal, alokasi tugas, dan evaluasi kinerja yang lebih baik.
- 2) **Manajemen Pasien:** Informasi pasien termasuk identitas pasien, riwayat pemeriksaan, dan detail kontak. Pentingnya manajemen pasien yang baik meliputi pencatatan riwayat medis, jadwal perawatan, dan integrasi dengan sistem pemeriksaan dan obat-obatan.
- 3) **Manajemen Periksa:** Data pemeriksaan mencakup kode pemeriksaan, obat-obatan yang digunakan, jumlah obat, dan total pembayaran. Dengan pengelolaan yang tepat, informasi ini dapat membantu dalam perencanaan penggunaan sumber daya dan pemantauan biaya operasional rumah sakit.
- 4) **Manajemen Obat-obatan:** Informasi tentang obat-obatan mencakup detail obat, stok, dan pemakaian. Sistem yang baik akan memastikan ketersediaan obat yang tepat waktu dan pencatatan yang akurat terhadap penggunaannya.
- 5) **Manajemen Pembayaran:** Pembayaran pasien dilacak dengan detail total pembayaran, metode pembayaran, dan tanggal pembayaran. Pengelolaan yang efisien terhadap informasi ini memungkinkan administrasi keuangan yang lebih baik dan audit yang akurat.

II. Deskripsi Tugas Akhir

Rumah sakit sebagai lembaga pelayanan kesehatan memiliki tanggung jawab besar dalam memberikan layanan medis kepada masyarakat. Dengan perkembangan teknologi informasi, penggunaan sistem database yang efektif dapat membantu rumah sakit dalam mengelola data pasien, dokter, pemeriksaan, obat, dan berbagai informasi penting lainnya. Database yang baik dapat meningkatkan efisiensi operasional, mengurangi kesalahan manusia, dan mempercepat pengambilan Keputusan.

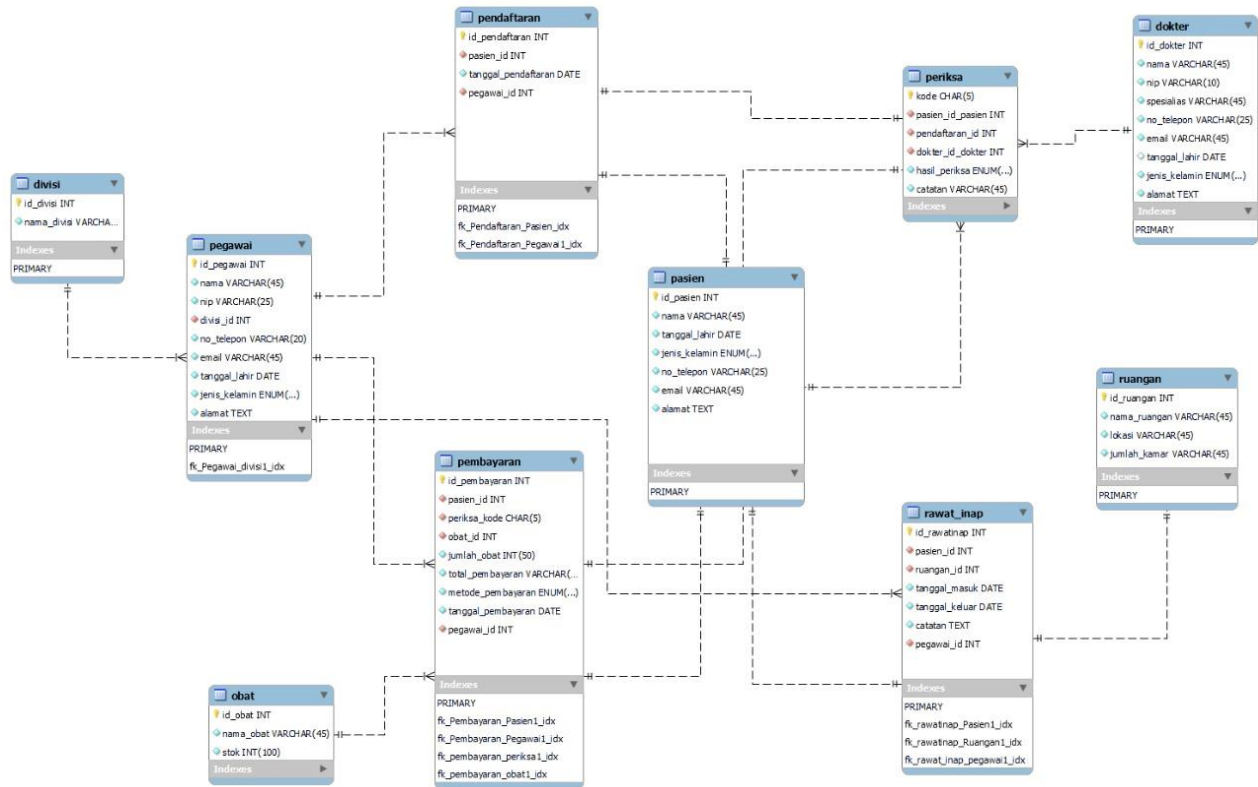
Tujuan dari proyek ini adalah untuk merancang dan mengimplementasikan sistem database yang terintegrasi dan efisien untuk mendukung operasional rumah sakit. Sistem ini diharapkan dapat mengelola berbagai jenis data yang ada di rumah sakit secara efektif dan memberikan kemudahan dalam pengolahan serta akses data.

Teknologi yang Digunakan:

- **MariaDB:** Sebagai sistem manajemen basis data relasional (RDBMS) yang handal dan scalable untuk menyimpan dan mengelola semua data rumah sakit.
- **Stored Procedure:** Dibuat untuk operasi yang kompleks dan terstruktur, seperti penambahan data pegawai, memastikan konsistensi dan keamanan dalam pengelolaan data.

- Views: Digunakan untuk menyediakan perspektif yang terstruktur terhadap data seperti detail pegawai, pasien perempuan, dan statistik pegawai berdasarkan divisi.
- Transaction Management: Digunakan untuk memastikan operasi database yang aman dan konsisten, terutama dalam transaksi yang melibatkan pembayaran pasien.

III. Desain Database dengan mysql workbench



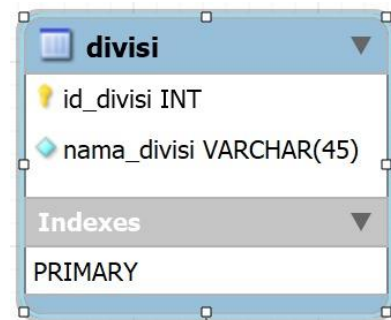
IV. Struktur Database

1) Tabel Divisi

Kolom :

- id_divisi : Tipe data integer, primary key, auto increment, not null.
- nama_divisi : Tipe data varchar untuk menyimpan nama divisi.

Deskripsi : Untuk menyimpan nama pegawai yang ada di rumah sakit.

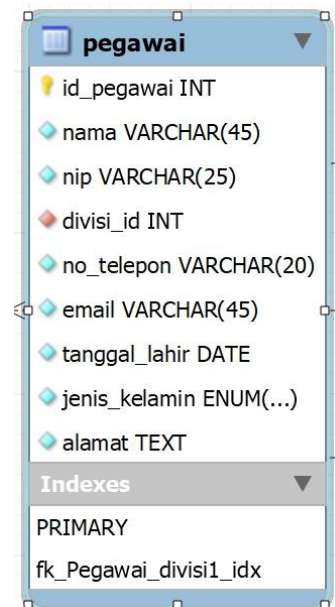


2) Tabel Pegawai

Kolom :

- id_pegawai : Tipe data integer, primary key, auto increment
- nama : Tipe data varchar untuk menyimpan nama pegawai.
- nip : Tipe data varchar untuk menyimpan nip pegawai.
- divisi_id : Tipe data integer untuk memanggil dan menyimpan id divisi pegawai yang sudah ditentukan.
- no_telepon : Tipe data varchar untuk menyimpan nomer telepon pegawai.
- email : Tipe data varchar untuk menyimpan email pegawai.
- tanggal_lahir : Tipe data date untuk menyimpan tanggal lahir pegawai.
- jenis_kelamin : Tipe data enum ('Laki-laki' 'Perempuan') untuk memilih jenis kelamin pegawai dan menyimpannya.
- alamat : Tipe data text untuk menyimpan alamat pegawai.

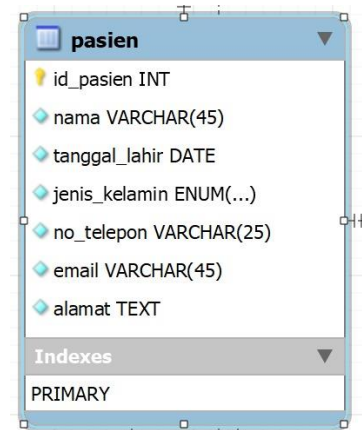
Deskripsi : Untuk menyimpan data pegawai yang ada di rumah sakit.



3) Tabel Pasien

Kolom :

- id_pasien Tipe data integer, primary key, auto increment, not null.
- Nama : Tipe data Varchar untuk menyimpan nama pasien
- Tanggal_lahir : Tipe data Date untuk menyimpan tanggal lahir pasien
- jenis_kelamin : Tipe data enum ('Laki-laki' 'Perempuan') untuk memilih jenis kelamin pegawai dan menyimpannya.
- no_telepon : Tipe data Varchar untuk menyimpan nomor telepon pasien
- email : Tipe data Varchar untuk menyimpan data email pasien
- alamat : Tipe data Text untuk menyimpan alamat pasien



Deskripsi : Untuk menyimpan data pasien yang hendak berobat atau di rawat dirumah sakit

4) Tabel Dokter

Kolom :

- id_dokter : Tipe data integer, primary key, auto increment
- nama : Tipe data varchar untuk menyimpan nama dokter.
- nip : Tipe data varchar untuk menyimpan nip dokter.
- spesialisasi : Tipe data integer untuk menyimpan bagian spesialisasi dokter yang sudah ditentukan.
- no_telepon : Tipe data varchar untuk menyimpan nomor telepon dokter.
- email : Tipe data varchar untuk menyimpan email dokter.
- tanggal_lahir : Tipe data date untuk menyimpan tanggal lahir dokter.
- jenis_kelamin : Tipe data enum ('Laki-laki' 'Perempuan') untuk memilih jenis kelamin dokter dan menyimpannya.
- alamat : Tipe data text untuk menyimpan alamat dokter.



Deskripsi : Untuk menyimpan data dokter yang melakukan pemeriksaan terhadap pasien yang sudah melakukan pendaftaran

5) Tabel Pendaftaran

Kolom:

- id_pendaftaran : Tipe data integer, primary key, auto increment
- pasien_id : Tipe data Integer untuk memanggil dan menyimpan id pasien
- tanggal_pendaftaran : Tipe data Date untuk menyimpan tanggal pendaftaran pasien
- pegawai_id : Tipe data Integer untuk memanggil dan menyimpan id pegawai

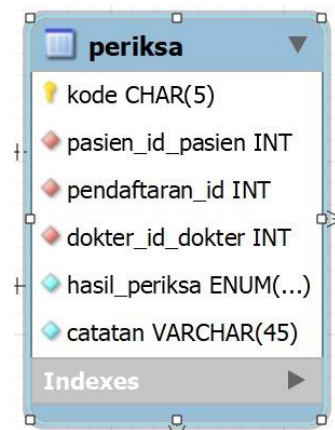
Deskripsi : Untuk menyimpan data pasien sebelum melakukan pemeriksaan dokter



6) Tabel Periksa

Kolom :

- Kode : Tipe data char untuk menyimpan kode periksa.
- pasien_id_pasien : Tipe data integer untuk memanggil dan menyimpan id pasien.
- pendaftaran_id : Tipe data integer untuk memanggil dan menyimpan id pendaftaran.
- dokter_id : Tipe data integer untuk memanggil dan menyimpan id dokter.
- hasil_periksa : Tipe data enum ('Tidak dirawat' 'Rawat') untuk memilih dan menyimpan hasil pemeriksaan dari pasien tersebut.
- catatan : Tipe data varchar untuk menyimpan catatan atau keterangan terkait hasil pemeriksaan.



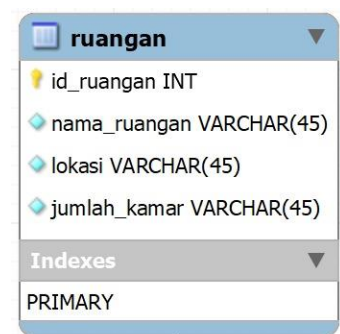
Deskripsi : Untuk menyimpan data hasil pemeriksaan pasien yang sudah diperiksa oleh dokter

7) Tabel Ruangan

Kolom :

- id_ruangan : Tipe data integer, primary key, auto increment.
- nama_ruangan : Tipe data varchar untuk menyimpan nama ruangan.
- lokasi : Tipe data varchar untuk menyimpan lokasi ruangan.
- jumlah_kamar : Tipe data varchar untuk menyimpan jumlah kamar yang tersedia yang ada di ruangan tersebut.

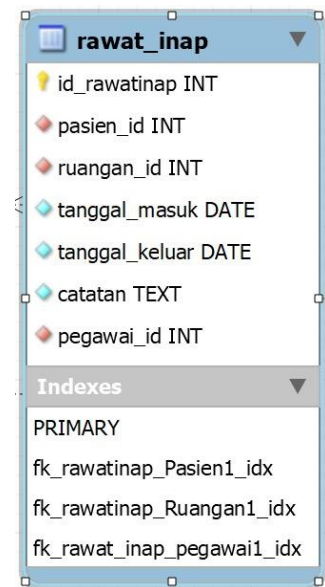
Deskripsi : Untuk menyimpan jumlah data kamar yang tersedia



8) Tabel Rawat Inap

Kolom:

- id_rawatinap : Tipe data integer, primary key, auto increment.
- pasien_id : Tipe data Integer untuk memanggil dan menyimpan id pasien.
- ruangan_id : Tipe data Integer untuk memanggil dan menyimpan id ruangan.
- tanggal_masuk : Tipe data Date untuk menyimpan data tanggal masuk pasien yang sudah ditentukan oleh dokter
- tanggal_keluar : Tipe data Date untuk menyimpan data tanggal keluar pasien yang sudah ditentukan oleh dokter
- catatan : Tipe data Text untuk menyimpan data catatan penyakit pasien
- pegawai_id : Tipe data Integer untuk memanggil dan menyimpan id pegawai



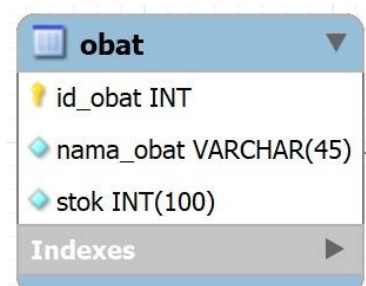
rawat_inap	
id_rawatinap	INT
pasien_id	INT
ruangan_id	INT
tanggal_masuk	DATE
tanggal_keluar	DATE
catatan	TEXT
pegawai_id	INT
Indexes	
PRIMARY	
fk_rawatinap_Pasien1_idx	
fk_rawatinap_Ruangan1_idx	
fk_rawat_inap_pegawai1_idx	

Deskripsi : Untuk menyimpan informasi data pasien yang hendak dirawat

9) Tabel Obat

Kolom:

- id_obat : Tipe data integer, primary key, auto increment.
- nama_obat : Tipe data varchar untuk menyimpan nama obat.
- stok : Tipe data integer untuk menyimpan data stok obat yang tersedia di Rumah sakit.



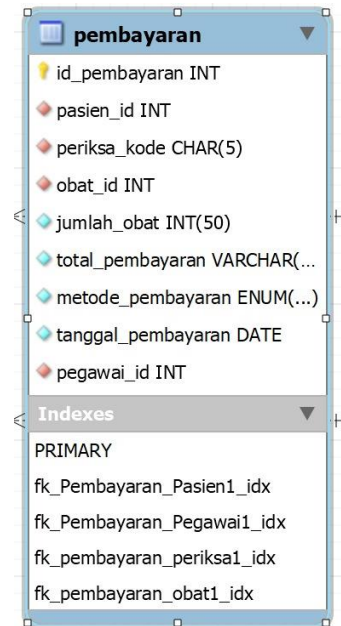
obat	
id_obat	INT
nama_obat	VARCHAR(45)
stok	INT(100)
Indexes	

Deskripsi : Untuk menyimpan informasi data obat yaitu nama obat dan stoknya yang ada di Rumah sakit.

10) Tabel Pembayaran

Kolom :

- id_pembayaran : Tipe data integer, primary key, auto increment.
- pasien_id : Tipe data Integer untuk memanggil dan menyimpan id pasien.
- periksa_kode : Tipe data Integer untuk memanggil dan menyimpan kode pemeriksaan pasien.
- obat_id : Tipe data Integer untuk memanggil dan menyimpan id obat yang diperoleh dari hasil pemeriksaan
- jumlah_obat : Tipe data Integer untuk menyimpan data jumlah obat yang dibutuhkan
- total_pembayaran : Tipe data Varchar untuk menyimpan data pembayaran pasien yang sudah melakukan pemeriksaan di rawat ataupun tidak dirawat
- metode_pembayaran : Tipe data Enum ('Tunai', 'Non-Tunai') untuk memilih dan menyimpan data pembayaran yang dilakukan oleh pasien
- tanggal_pembayaran : Tipe data Date untuk menyimpan data tanggal pembayaran yang dilakukan oleh pasien
- pegawai_id : Tipe data Integer untuk memanggil dan menyimpan id pegawai



Deskripsi : Untuk menyimpan informasi data pembayaran yang berisikan data pasien dan obat.

V. Relasi Table

➤ Tabel divisi

Hubungan one-to-many (satu ke banyak) antara divisi dan pegawai menggambarkan bahwa satu divisi dapat memiliki banyak pegawai, tetapi setiap pegawai hanya dapat berada di satu divisi. Ini adalah konsep umum dalam struktur organisasi di mana divisi atau departemen bertanggung jawab atas sejumlah pegawai.

➤ Tabel Pegawai

Hubungan one-to-many (satu ke banyak) antara pegawai dan entitas lain seperti pembayaran, pendaftaran dan rawat inap berarti bahwa satu pegawai dapat memiliki banyak entri pembayaran, pendaftaran dan rawat inap, tetapi setiap entri pembayaran, pendaftaran dan rawat inap hanya terkait dengan satu pegawai.

Berikut penjelasan lebih lanjut:

- **Pendaftaran:** Setiap pegawai bisa memiliki banyak entri pendaftaran, namun setiap entri pendaftaran hanya terkait dengan satu pegawai.
- **Pembayaran:** Setiap pegawai bisa memiliki banyak entri pembayaran, namun setiap entri pembayaran hanya terkait dengan satu pegawai.
- **Rawat Inap:** Setiap pegawai bisa memiliki banyak entri rawat inap, namun setiap entri rawat inap hanya terkait dengan satu pegawai

➤ Tabel Pasien

Hubungan one-to-one antara pasien dan entitas lain seperti pembayaran, pendaftaran, dan rawat inap memastikan bahwa setiap pasien memiliki satu catatan terkait, dan sebaliknya. Ini berarti:

- **Pembayaran:** Setiap pasien memiliki satu entri pembayaran terkait, memastikan pembayaran dicatat sekali untuk setiap pasien.
- **Pendaftaran:** Setiap pasien memiliki satu entri pendaftaran terkait, memastikan pendaftaran dicatat khusus untuk setiap pasien.
- **Rawat Inap:** Setiap pasien memiliki satu entri rawat inap terkait, memastikan rawat inap dicatat satu kali untuk setiap pasien.

Hubungan many-to-many antara pasien dan dokter memungkinkan fleksibilitas dalam pengelolaan data medis, di mana setiap pasien bisa ditangani oleh banyak dokter dan setiap dokter bisa menangani banyak pasien. Tabel penghubung "Periksa" digunakan untuk mencatat detail setiap interaksi, memastikan data tersimpan dengan baik dan dapat diakses dengan mudah.

➤ Tabel Dokter

Hubungan many-to-many (banyak ke banyak) antara dokter dan pasien menggambarkan bahwa setiap dokter dapat merawat banyak pasien, dan setiap pasien dapat dirawat oleh banyak dokter. Untuk mengimplementasikan hubungan ini dalam basis data, digunakan tabel penghubung (junction table) yang disebut "Periksa".

➤ Tabel Periksa

Hubungan one-to-one (satu ke satu) antara "Periksa" dan entitas lain seperti "Pembayaran" dan "Pendaftaran" menggambarkan bahwa setiap entri dalam tabel "Periksa" hanya memiliki satu entri terkait dalam tabel "Pembayaran" dan satu entri terkait dalam tabel "Pendaftaran", dan sebaliknya. Tabel "Periksa" sendiri adalah hasil dari hubungan many-to-many antara "Pasien" dan "Dokter".

- **Pembayaran:** Setiap entri pemeriksaan (Periksa) memiliki satu entri pembayaran yang terkait, dan setiap entri pembayaran hanya terkait dengan satu pemeriksaan.
- **Pendaftaran:** Setiap entri pemeriksaan (Periksa) memiliki satu entri pendaftaran yang terkait, dan setiap entri pendaftaran hanya terkait dengan satu pemeriksaan.

➤ Tabel Pendaftaran

Hubungan one-to-one (satu ke satu) antara "Pendaftaran" dan entitas lain seperti "Periksa" dan "Pasien" menggambarkan bahwa setiap entri dalam tabel "Pendaftaran" hanya memiliki satu entri terkait dalam tabel "periksa" dan satu entri terkait dalam tabel "Pasien", dan sebaliknya.

- **Periksa:** setiap pendaftaran terkait dengan satu pemeriksaan, dan setiap pemeriksaan terkait dengan satu pendaftaran.
- **Pasien:** setiap pendaftaran terkait dengan satu pasien, dan setiap pasien terkait dengan satu pendaftaran.

➤ Tabel Ruangan

Hubungan one-to-one antara tabel "Ruangan" dan tabel "Rawat Inap" berarti setiap entri dalam tabel "Ruangan" hanya terkait dengan satu entri dalam tabel "Rawat Inap", dan sebaliknya. Ini mengindikasikan bahwa setiap ruangan hanya digunakan untuk satu pasien dalam satu periode waktu tertentu, dan setiap pasien hanya ditempatkan di satu ruangan. Dalam konteks perawatan medis, hal ini menunjukkan bahwa setiap pasien dirawat di satu ruangan khusus.

➤ Tabel Rawat Inap

Hubungan one-two-one antara tabel "Pasien" dan tabel "Ruangan" berikut penjelasannya:

- **Rawat Inap ke Pasien:** Setiap entri dalam tabel "Rawat Inap" hanya terkait dengan satu entri dalam tabel "Pasien", dan sebaliknya. Ini menunjukkan bahwa setiap rawat inap hanya terkait dengan satu pasien, dan setiap pasien hanya memiliki satu rawat inap.
- **Rawat Inap ke Ruangan:** Setiap entri dalam tabel "Rawat Inap" hanya terkait dengan satu entri dalam tabel "Ruangan", dan sebaliknya. Ini menunjukkan bahwa setiap rawat inap terkait dengan satu ruangan, dan setiap ruangan hanya digunakan untuk satu rawat inap pada suatu waktu tertentu.

➤ **Tabel Obat**

Hubungan one-to-many (satu ke banyak) antara tabel "Obat" dan tabel "Pembayaran" berarti Setiap entri dalam tabel "Obat" dapat terkait dengan banyak entri dalam tabel "Pembayaran". Namun, setiap entri dalam tabel "Pembayaran" hanya terkait dengan satu entri dalam tabel "Obat".

Ini menunjukkan bahwa satu transaksi pembayaran dapat melibatkan pembelian banyak jenis obat yang berbeda, tetapi setiap jenis obat hanya terkait dengan satu transaksi pembayaran.

➤ **Tabel Pembayaran**

Hubungan one-to-one (satu ke satu) antara tabel "Pembayaran" dan tabel "Pasien", serta tabel "Pembayaran" dan tabel "Periksa" berarti:

- Pembayaran ke Pasien: Setiap entri dalam tabel "Pembayaran" hanya terkait dengan satu entri dalam tabel "Pasien", dan sebaliknya. Ini menunjukkan bahwa setiap pembayaran hanya terkait dengan satu pasien, dan setiap pasien hanya memiliki satu entri pembayaran.
- Pembayaran ke Periksa: Setiap entri dalam tabel "Pembayaran" hanya terkait dengan satu entri dalam tabel "Periksa", dan sebaliknya. Ini menunjukkan bahwa setiap pembayaran hanya terkait dengan satu entri pemeriksaan, dan setiap pemeriksaan hanya memiliki satu entri pembayaran.

VI. Pembahasan Query

a. CRUD

```
MariaDB [(none)]> USE rumahsakit;
Database changed
MariaDB [rumahsakit]> INSERT INTO pegawai (nama, nip, divisi_id, no_telepon, email, tanggal_lahir, jenis_kelamin, alamat) VALUES
-> ('Rizky Putra', '678912345', 3, '081213140809', 'rizky@gmail.com', '1998-02-23', 'Laki-laki', 'Jl. Mangga Dua No. 20, Depok'),
-> ('Maharani Tista', '789123456', 3, '081213140810', 'rani@gmail.com', '1996-05-13', 'Perempuan', 'Jl. Diponegoro No. 5, Depok');
Query OK, 2 rows affected (0.005 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
MariaDB [rumahsakit]> SELECT * FROM pegawai WHERE id_pegawai = 31;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id_pegawai | nama      | nip      | divisi_id | no_telepon | email      | tanggal_lahir | jenis_kelamin | alamat      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 31         | Rizky Putra | 678912345 | 3         | 081213140809 | rizky@gmail.com | 1998-02-23    | Laki-laki    | Jl. Mangga Dua No. 20, Depok |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.002 sec)
```

```
MariaDB [rumahsakit]> UPDATE pegawai
-> SET email = 'maharani@gmail.com', tanggal_lahir = '1995-05-25', alamat = 'Jl. Surya Kencana No. 8, Depok'
-> WHERE id_pegawai = 32;
Query OK, 1 row affected (0.003 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
MariaDB [rumahsakit]> DELETE FROM pegawai WHERE id_pegawai = 32;
Query OK, 1 row affected (0.004 sec)
```

CREATE :

Perintah INSERT INTO digunakan untuk menambahkan data baru ke dalam tabel. Dalam hal ini, 2 data baru ditambahkan ke dalam tabel pegawai.

READ :

Perintah SQL ini digunakan untuk mengambil semua informasi yang terkait dengan pegawai yang memiliki id_pegawai 31 dari tabel pegawai. Hasilnya menampilkan semua kolom yang tersedia untuk baris yang sesuai dengan kondisi WHERE.

UPDATE :

Perintah Kueri ini digunakan untuk memperbarui data yang ingin diedit seperti kolom email, no_telepon, dan alamat yang sesuai pada tabel pegawai dan mengaksesnya dengan kondisi WHERE id_pegawai 32.

DELETE :

Perintah kueri ini menggunakan pernyataan DELETE untuk menghapus data pada baris tertentu dari tabel pegawai tepatnya pada baris yang mempunyai id_pegawai 32 dengan menggunakan kondisi WHERE.

b. INNER JOIN

```
MariaDB [rumahsakit]> SELECT pegawai.id_pegawai, pegawai.nama, pegawai.nip, divisi.nama_divisi
-> FROM pegawai
-> INNER JOIN divisi ON pegawai.divisi_id = divisi.id_divisi;
```

id_pegawai	nama	nip	nama_divisi
1	John Doe	123456789	Administrasi Pendaftaran
2	Jane Doe	234567890	Administrasi Pendaftaran
3	Michael Smith	345678901	Administrasi Pendaftaran
4	Emily Johnson	456789012	Administrasi Pendaftaran
5	David Lee	567890123	Administrasi Pendaftaran
6	Sarah Brown	678901234	Administrasi Pendaftaran
7	Muhammad Rahman	789012345	Administrasi Pendaftaran
8	Siti Rahayu	890123456	Administrasi Pendaftaran
9	Kevin Tan	901234567	Administrasi Pendaftaran
10	Ayu Wulandari	012345678	Administrasi Pendaftaran
11	Ahmad Hakim	123456780	Administrasi Pembayaran
12	Lina Kartika	234567801	Administrasi Pembayaran
13	Budi Santoso	345678012	Administrasi Pembayaran
14	Cinta Dewi	456780123	Administrasi Pembayaran
15	Eko Prasetyo	567801234	Administrasi Pembayaran
16	Sari Fitri	678012345	Administrasi Pembayaran
17	Joko Susilo	789012356	Administrasi Pembayaran
18	Rina Handayani	890123567	Administrasi Pembayaran
19	Agus Setiawan	901234678	Administrasi Pembayaran
20	Maya Sari	012345789	Administrasi Pembayaran
21	Anton Widodo	123456780	Manajemen Rawat Inap
22	Susi Rahayu	234567801	Manajemen Rawat Inap
23	Dodi Pramono	345678012	Manajemen Rawat Inap
24	Yuli Cahyani	456780123	Manajemen Rawat Inap
25	Rudi Setiawan	567801234	Manajemen Rawat Inap
26	Dewi Susanti	678012345	Manajemen Rawat Inap
27	Toni Pratama	789012356	Manajemen Rawat Inap
28	Rina Wijaya	890123567	Manajemen Rawat Inap
29	Agus Suryanto	901234678	Manajemen Rawat Inap
30	Rika Anggraini	012345789	Manajemen Rawat Inap

30 rows in set (0.007 sec)

INNER JOIN adalah jenis join yang hanya mengembalikan baris-baris yang memiliki kecocokan di kedua tabel yang digabungkan. Dalam query di atas,

`SELECT pegawai.id_pegawai, pegawai.nama, pegawai.nip, divisi.nama_divisi` menentukan kolom-kolom yang akan ditampilkan, yaitu ID pegawai, nama pegawai, NIP pegawai, dan nama divisi tempat pegawai bekerja.

`Bagian FROM pegawai` menunjukkan bahwa tabel utama yang digunakan adalah tabel pegawai. `INNER JOIN divisi ON pegawai.divisi_id = divisi.id_divisi` menggabungkan tabel pegawai dengan tabel divisi berdasarkan kecocokan antara kolom `divisi_id` di tabel pegawai dan kolom `id_divisi` di tabel divisi.

Dengan demikian, query ini menghasilkan daftar pegawai beserta informasi divisi tempat mereka bekerja, tetapi hanya untuk pegawai yang memiliki kecocokan divisi di kedua tabel tersebut.

c. OUTER JOIN

```
MariaDB [rumahsakit]> SELECT pegawai.id_pegawai, pegawai.nama, pegawai.nip, divisi.nama_divisi, pegawai.no_telepon, pegawai.email
-> FROM pegawai
-> RIGHT OUTER JOIN divisi ON pegawai.divisi_id = divisi.id_divisi;
```

id_pegawai	nama	nip	nama_divisi	no_telepon	email
1	John Doe	123456789	Administrasi Pendaftaran	08123456789	johndoe@gmail.com
2	Jane Doe	234567890	Administrasi Pendaftaran	08234567890	janedoe@gmail.com
3	Michael Smith	345678901	Administrasi Pendaftaran	08345678901	michaelsmith@gmail.com
4	Emily Johnson	456789012	Administrasi Pendaftaran	08456789012	emilyjohnson@gmail.com
5	David Lee	567890123	Administrasi Pendaftaran	08567890123	davidlee@gmail.com
6	Sarah Brown	678901234	Administrasi Pendaftaran	08678901234	sarahbrown@gmail.com
7	Muhammad Rahman	789012345	Administrasi Pendaftaran	08789012345	rahman.m@gmail.com
8	Siti Rahayu	890123456	Administrasi Pendaftaran	087901234567	siti.rahayu@gmail.com
9	Kevin Tan	901234567	Administrasi Pendaftaran	08890123456	kevin.tan@gmail.com
10	Ayu Wulandari	012345678	Administrasi Pendaftaran	08133456789	ayuwulan@gmail.com
11	Ahmad Hakim	123456780	Administrasi Pembayaran	08123456780	ahmad@gmail.com
12	Lina Kartika	234567801	Administrasi Pembayaran	08234567801	lina@gmail.com
13	Budi Santoso	345678012	Administrasi Pembayaran	08345678012	budi@gmail.com
14	Cinta Dewi	456780123	Administrasi Pembayaran	08456780123	cinta@gmail.com
15	Eko Prasetyo	567801234	Administrasi Pembayaran	08567801234	eko@gmail.com
16	Sari Fitri	678012345	Administrasi Pembayaran	08678012345	sari@gmail.com
17	Joko Susilo	789012356	Administrasi Pembayaran	08790123456	joko@gmail.com
18	Rina Handayani	890123567	Administrasi Pembayaran	08901235678	rina@gmail.com
19	Agus Susilo	901234678	Administrasi Pembayaran	08012346789	agus@gmail.com
20	Maya Sari	012345789	Administrasi Pembayaran	08123457890	maya@gmail.com
21	Anton Widodo	123456780	Manajemen Rawat Inap	08123456780	anton@gmail.com
22	Susi Rahayu	234567801	Manajemen Rawat Inap	08234567801	susi@gmail.com
23	Dodi Pramono	345678012	Manajemen Rawat Inap	08345678012	dodi@gmail.com
24	Yuli Cahyani	456780123	Manajemen Rawat Inap	08456780123	yuli@gmail.com
25	Rudi Setiawan	567801234	Manajemen Rawat Inap	08567801234	rudi@gmail.com
26	Dewi Susanti	678012345	Manajemen Rawat Inap	08678012345	dewi@gmail.com
27	Toni Pratama	789012356	Manajemen Rawat Inap	08790123456	toni@gmail.com
28	Rina Wijaya	890123567	Manajemen Rawat Inap	08901235678	rina@gmail.com
29	Agus Suryanto	901234678	Manajemen Rawat Inap	080126789	agus@gmail.com
30	Rika Anggraini	012345789	Manajemen Rawat Inap	08123457890	rika@gmail.com

30 rows in set (0.001 sec)

Outer Join adalah tipe join dalam SQL yang mengembalikan semua baris dari satu tabel dan baris yang cocok dari tabel lainnya, dengan nilai NULL untuk kolom yang tidak memiliki kecocokan.

Dalam query yang di atas, kita menggunakan Right Outer Join antara tabel `pegawai` dan `divisi`, dimana `divisi` adalah tabel kanan dan `pegawai` adalah tabel kiri. Kondisi join-nya adalah `pegawai.divisi_id = divisi.id_divisi`.

Query ini akan mengembalikan semua baris dari tabel `divisi` dan baris yang cocok dari tabel `pegawai`. Jika tidak ada kecocokan di tabel `pegawai`, kolom yang berasal dari `pegawai` akan berisi NULL.

Sebagai contoh, jika ada divisi yang tidak memiliki pegawai yang terdaftar, hasil query akan menunjukkan informasi divisi tersebut dengan kolom pegawai yang berisi NULL. Ini berguna untuk memastikan bahwa semua divisi terdaftar muncul dalam hasil query, meskipun tidak ada pegawai yang terhubung dengan divisi tersebut.

d. INDEX

```
MariaDB [rumahsakit]> CREATE INDEX idx_nama ON pegawai(nama);
Query OK, 0 rows affected (0.016 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> SHOW CREATE TABLE pegawai;
+-----+
| Table | Create Table
+-----+
| pegawai | CREATE TABLE 'pegawai' (
  'id_pegawai' int(11) NOT NULL AUTO_INCREMENT,
  'nama' varchar(45) NOT NULL,
  'nip' varchar(25) NOT NULL,
  'divisi_id' int(11) NOT NULL,
  'no_telepon' varchar(20) NOT NULL,
  'email' varchar(45) NOT NULL,
  'tanggal_lahir' date NOT NULL,
  'jenis_kelamin' enum('Laki-laki','Perempuan') NOT NULL,
  'alamat' text NOT NULL,
  PRIMARY KEY ('id_pegawai'),
  KEY 'fk_Pegawai_divisi1_idx' ('divisi_id'),
  KEY 'idx_nama' ('nama'),
  CONSTRAINT 'fk_Pegawai_divisi1' FOREIGN KEY ('divisi_id') REFERENCES 'divisi' ('id_divisi') ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci |
+-----+

1 row in set (0.000 sec)
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> SHOW INDEX FROM pegawai;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| pegawai | 0 | PRIMARY | 1 | id_pegawai | A | 30 | NULL | NULL | NULL | BTREE | | |
| pegawai | 1 | fk_Pegawai_divisi1_idx | 1 | divisi_id | A | 6 | NULL | NULL | NULL | BTREE | | |
| pegawai | 1 | idx_nama | 1 | nama | A | 30 | NULL | NULL | NULL | BTREE | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.001 sec)
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> SELECT * FROM pegawai WHERE nama = 'Sarah Brown';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id_pegawai | nama | nip | divisi_id | no_telepon | email | tanggal_lahir | jenis_kelamin | alamat |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 6 | Sarah Brown | 678901234 | 1 | 08678901234 | sarahbrown@gmail.com | 1997-04-18 | Perempuan | Jl. HR Rasuna Said No. 34, Jakarta |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> SELECT * FROM pegawai ORDER BY nama;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id_pegawai | nama | nip | divisi_id | no_telepon | email | tanggal_lahir | jenis_kelamin | alamat |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 19 | Agus Setiawan | 901234678 | 2 | 080123456789 | agus@gmail.com | 1995-05-15 | Laki-laki | Jl. Sudirman Kav. 25, Jakarta |
| 20 | Agus Suryanto | 901234678 | 3 | 080126789 | agus@gmail.com | 1996-06-15 | Laki-laki | Jl. Jenderal Sudirman No. 29, Jakarta |
| 11 | Ahmad Hakim | 123456780 | 2 | 08123456780 | ahmad@gmail.com | 1993-01-05 | Laki-laki | Jl. Tebet Barat No. 5, Jakarta |
| 21 | Anton Widodo | 123456780 | 3 | 08123456780 | anton@gmail.com | 1990-06-15 | Laki-laki | Jl. Sudirman No. 21, Jakarta |
| 10 | Ayu Wulandari | 012345678 | 1 | 08133456789 | ayuwulandari@gmail.com | 1996-03-08 | Perempuan | Jl. MT Haryono No. 30, Jakarta |
| 13 | Budi Santoso | 345678012 | 2 | 08345678012 | budi@gmail.com | 1997-11-15 | Laki-laki | Jl. Pangeran Antasari No. 23, Jakarta |
| 14 | Cinta Dewi | 456789123 | 2 | 08456789123 | cinta@gmail.com | 1992-04-20 | Perempuan | Jl. Rasuna Said Kav. 20, Jakarta |
| 5 | David Lee | 567890123 | 1 | 08567890123 | davidlee@gmail.com | 1983-07-25 | Laki-laki | Jl. MH Thamrin No. 80, Jakarta |
| 26 | Dewi Susanti | 678901234 | 3 | 08678901234 | dewi@gmail.com | 1989-04-18 | Perempuan | Jl. Jenderal Sudirman No. 26, Jakarta |
| 23 | Dodi Pramono | 345678012 | 3 | 08345678012 | dodi@gmail.com | 1984-05-25 | Laki-laki | Jl. Gatot Subroto No. 23, Jakarta |
| 15 | Eko Prasetyo | 567891234 | 2 | 08567890123 | eko@gmail.com | 1985-09-25 | Laki-laki | Jl. Cikini Raya No. 8, Jakarta |
| 4 | Emily Johnson | 456789012 | 1 | 08456789012 | emilyjohnson@gmail.com | 1997-11-30 | Perempuan | Jl. Kuningan No. 12, Jakarta |
| 2 | Jane Doe | 234567890 | 1 | 08234567890 | janedoe@gmail.com | 1992-08-20 | Perempuan | Jl. Gatot Subroto No. 45, Jakarta |
| 1 | John Doe | 123456789 | 2 | 08123456789 | johndoe@gmail.com | 1990-05-15 | Laki-laki | Jl. Sudirman No. 123, Jakarta |
| 17 | Joko Susilo | 789012356 | 2 | 08790123456 | joko@gmail.com | 1991-03-12 | Laki-laki | Jl. Tanah Abang No. 15, Jakarta |
| 9 | Kevin Tan | 901234567 | 1 | 08090123456 | kevin.tan@gmail.com | 1991-06-20 | Laki-laki | Jl. Gatot Subroto Kav. 18, Jakarta |
| 12 | Lina Kartika | 234567801 | 2 | 08234567801 | lina@gmail.com | 1990-07-10 | Perempuan | Jl. Cipinang Compedak No. 17, Jakarta |
| 20 | Maya Sari | 012345789 | 2 | 08123457890 | maya@gmail.com | 1989-10-22 | Perempuan | Jl. Hayan Wuruk No. 19, Jakarta |
| 3 | Michael Smith | 345678901 | 1 | 08345678901 | michaelsmith@gmail.com | 1995-02-10 | Laki-laki | Jl. Thamrin No. 67, Jakarta |
| 7 | Muhammad Rahman | 789012345 | 1 | 08789012345 | rahman.m@gmail.com | 1995-09-05 | Laki-laki | Jl. HR Rasuna Said No. 34, Jakarta |
| 28 | Rika Anggraini | 012345789 | 3 | 08123457890 | rika@gmail.com | 1990-09-22 | Perempuan | Jl. MH Thamrin No. 30, Jakarta |
| 18 | Rina Handayani | 890123567 | 2 | 08901235678 | rina@gmail.com | 1983-08-05 | Perempuan | Jl. Kebon Sirih No. 22, Jakarta |
| 28 | Rina Wijaya | 890123567 | 3 | 08901235678 | rina@gmail.com | 1984-08-05 | Perempuan | Jl. Jenderal Gatot Subroto No. 28, Jakarta |
| 25 | Rudi Setiawan | 567890123 | 3 | 08567890123 | rudi@gmail.com | 1986-07-30 | Laki-laki | Jl. MH Thamrin No. 25, Jakarta |
| 6 | Sarah Brown | 678901234 | 1 | 08678901234 | sarahbrown@gmail.com | 1997-04-18 | Perempuan | Jl. HR Rasuna Said No. 34, Jakarta |
| 16 | Sari Fitri | 6789012345 | 2 | 086789012345 | sari@gmail.com | 1988-06-30 | Perempuan | Jl. Gatot Subroto No. 67, Jakarta |
| 8 | Siti Rahayu | 890123456 | 1 | 08790123456 | siti.rahayu@gmail.com | 1993-12-10 | Perempuan | Jl. Palmerah Barat No. 70, Jakarta |
| 22 | Susi Rahayu | 234567801 | 3 | 08234567801 | susi@gmail.com | 1987-12-10 | Perempuan | Jl. Thamrin No. 22, Jakarta |
| 27 | Toni Pratama | 789012356 | 3 | 08790123456 | toni@gmail.com | 1992-11-22 | Laki-laki | Jl. HR Rasuna Said No. 27, Jakarta |
| 24 | Yuli Cahyani | 4567890123 | 3 | 084567890123 | yuli@gmail.com | 1993-02-20 | Perempuan | Jl. Rasuna Said No. 24, Jakarta |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
30 rows in set (0.001 sec)
MariaDB [rumahsakit]> |
```


Indeks dalam basis data adalah struktur data yang meningkatkan kecepatan pengambilan baris dari tabel pada kolom tertentu. Ketika indeks dibuat pada kolom tertentu, basis data dapat menemukan baris dengan nilai yang cocok jauh lebih cepat dibandingkan dengan pencarian tanpa indeks.

Dalam konteks query di atas, indeks `idx_nama` dibuat pada kolom `nama` dari tabel `pegawai`. Hal ini memungkinkan query seperti `SELECT * FROM pegawai WHERE nama = 'Sarah Brown';` untuk dieksekusi lebih cepat, karena basis data dapat menggunakan indeks untuk langsung menemukan baris yang sesuai dengan nama yang dicari, tanpa harus memindai seluruh tabel.

Selain itu, indeks juga dapat mempercepat query yang mengurutkan hasil berdasarkan kolom yang diindeks, seperti `SELECT * FROM pegawai ORDER BY nama;`, karena basis data dapat menggunakan urutan yang sudah terindeks alih-alih menyortir data secara manual setiap kali query dijalankan. Dengan demikian, indeks pada kolom `nama` secara signifikan meningkatkan efisiensi eksekusi query yang melibatkan pencarian atau pengurutan berdasarkan nama pegawai.

e. VIEW

View dalam basis data adalah objek virtual yang menyajikan hasil dari query SQL seolah-olah itu adalah tabel. View digunakan untuk menyederhanakan query kompleks, meningkatkan keamanan data dengan membatasi akses langsung ke tabel, dan menyajikan data dalam format yang lebih mudah dipahami.

```
MariaDB [rumahsakit]> CREATE VIEW view_pegawai_detail AS
-> SELECT
->     p.id_pegawai,
->     p.nama,
->     p.nip,
->     d.nama_divisi,
->     p.no_telepon,
->     p.email,
->     p.tanggal_lahir,
->     p.jenis_kelamin,
->     p.alamat
-> FROM
->     pegawai p
-> JOIN
->     divisi d ON p.divisi_id = d.id_divisi;
Query OK, 0 rows affected (0.006 sec)

MariaDB [rumahsakit]> |
```

```

MariaDB [rumahsakit]> CREATE VIEW view_pegawai_perempuan AS
-> SELECT
->     id_pegawai,
->     nama,
->     nip,
->     divisi_id,
->     no_telepon,
->     email,
->     tanggal_lahir,
->     alamat
-> FROM
->     pegawai
-> WHERE
->     jenis_kelamin = 'Perempuan';
Query OK, 0 rows affected (0.004 sec)

MariaDB [rumahsakit]> |

```

```

MariaDB [rumahsakit]> CREATE VIEW view_pegawai_by_divisi AS
-> SELECT
->     d.nama_divisi,
->     COUNT(p.id_pegawai) AS jumlah_pegawai
-> FROM
->     pegawai p
-> JOIN
->     divisi d ON p.divisi_id = d.id_divisi
-> GROUP BY
->     d.nama_divisi;
Query OK, 0 rows affected (0.004 sec)

MariaDB [rumahsakit]> |

```

Dalam konteks query di atas, tiga view dibuat untuk keperluan yang berbeda.

'view_pegawai_detail' menyajikan detail lengkap pegawai bersama nama divisinya dengan melakukan join antara tabel 'pegawai' dan 'divisi'. View ini memudahkan dalam melihat informasi lengkap pegawai termasuk nama divisi tanpa perlu melakukan join setiap kali query dijalankan.

'view_pegawai_perempuan' menyaring data pegawai untuk hanya menampilkan pegawai perempuan. Ini berguna untuk mendapatkan daftar pegawai perempuan dengan lebih cepat tanpa perlu menulis klausa WHERE setiap kali.

'view_pegawai_by_divisi' mengelompokkan pegawai berdasarkan divisinya dan menghitung jumlah pegawai di setiap divisi. Ini berguna untuk analisis cepat mengenai distribusi pegawai dalam berbagai divisi tanpa perlu menulis query pengelompokan (GROUP BY) setiap kali.

Dengan menggunakan view, pengguna dapat dengan mudah mengakses dan menganalisis data yang kompleks tanpa perlu menulis query panjang setiap kali, serta meningkatkan manajemen dan keamanan akses data.

f. TRANSACTION COMMIT

```
MariaDB [rumahsakit]> START TRANSACTION;
Query OK, 0 rows affected (0.000 sec)

MariaDB [rumahsakit]> |

MariaDB [rumahsakit]> INSERT INTO pembayaran (id_pembayaran, pasien_id, periksa_kode, obat_id, jumlah_obat, total_pembayaran, metode_pembayaran, tanggal_pembayaran, pegawai_id)
-> VALUES
-> (15,2,2,1,30000,'Non-Tunai','2024-06-19',13),
-> (12,16,6,3,90000,'Non-Tunai','2024-06-25',18);
Query OK, 2 rows affected (0.001 sec)
Records: 2 Duplicates: 0 Warnings: 0

MariaDB [rumahsakit]> COMMIT;
Query OK, 0 rows affected (0.003 sec)

MariaDB [rumahsakit]> |

MariaDB [rumahsakit]> SELECT * FROM pembayaran;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id_pembayaran | pasien_id | periksa_kode | obat_id | jumlah_obat | total_pembayaran | metode_pembayaran | tanggal_pembayaran | pegawai_id |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 12            | 16        | 16           | 6        | 3           | 90000            | Non-Tunai        | 2024-06-25         | 18         |
| 15            | 2         | 2            | 2        | 1           | 30000            | Non-Tunai        | 2024-06-19         | 13         |
| 21            | 1         | 1            | 1        | 2           | 60000            | Tunai            | 2024-06-01         | 12         |
| 22            | 2         | 2            | 2        | 1           | 30000            | Non-Tunai        | 2024-06-01         | 12         |
| 23            | 3         | 3            | 3        | 3           | 90000            | Non-Tunai        | 2024-06-02         | 13         |
| 24            | 4         | 4            | 4        | 2           | 60000            | Tunai            | 2024-06-02         | 13         |
| 25            | 5         | 5            | 5        | 1           | 30000            | Non-Tunai        | 2024-06-03         | 14         |
| 26            | 6         | 6            | 6        | 1           | 30000            | Non-Tunai        | 2024-06-03         | 14         |
| 27            | 7         | 7            | 7        | 2           | 60000            | Tunai            | 2024-06-04         | 15         |
| 28            | 8         | 8            | 8        | 1           | 30000            | Tunai            | 2024-06-04         | 15         |
| 29            | 9         | 9            | 9        | 3           | 90000            | Non-Tunai        | 2024-06-05         | 16         |
| 30            | 10        | 10           | 10       | 2           | 60000            | Tunai            | 2024-06-05         | 16         |
| 31            | 11        | 11           | 11       | 2           | 60000            | Tunai            | 2024-06-06         | 17         |
| 32            | 12        | 12           | 12       | 1           | 30000            | Non-Tunai        | 2024-06-06         | 17         |
| 33            | 13        | 13           | 13       | 3           | 90000            | Non-Tunai        | 2024-06-07         | 18         |
| 34            | 14        | 14           | 14       | 2           | 60000            | Tunai            | 2024-06-07         | 18         |
| 35            | 15        | 15           | 15       | 1           | 30000            | Non-Tunai        | 2024-06-08         | 19         |
| 36            | 16        | 16           | 16       | 3           | 90000            | Non-Tunai        | 2024-06-08         | 19         |
| 37            | 17        | 17           | 17       | 2           | 60000            | Tunai            | 2024-06-09         | 20         |
| 38            | 18        | 18           | 18       | 1           | 30000            | Non-Tunai        | 2024-06-09         | 20         |
| 39            | 19        | 19           | 19       | 3           | 90000            | Non-Tunai        | 2024-06-10         | 12         |
| 40            | 20        | 20           | 20       | 2           | 60000            | Tunai            | 2024-06-10         | 12         |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
22 rows in set (0.001 sec)

MariaDB [rumahsakit]> |
```

Transaction dalam basis data adalah serangkaian operasi yang dieksekusi sebagai unit yang atomik, yang berarti semua operasi dalam transaksi harus berhasil atau tidak ada yang dieksekusi sama sekali.

Komitmen transaksi (COMMIT) adalah perintah yang digunakan untuk menyimpan semua perubahan yang dibuat selama transaksi ke basis data secara permanen.

Dalam konteks query di atas, transaksi dimulai dengan perintah **'START TRANSACTION;'** dan mencakup dua perintah INSERT yang menambahkan data baru ke tabel 'pembayaran'. Setelah memastikan bahwa kedua operasi INSERT berhasil tanpa error, perintah **'COMMIT;'** dieksekusi untuk memastikan bahwa perubahan tersebut disimpan secara permanen di basis data.

Jika terjadi kegagalan atau error sebelum COMMIT dieksekusi, transaksi dapat dibatalkan dengan perintah **'ROLLBACK;'**, sehingga basis data dikembalikan ke kondisi semula sebelum transaksi dimulai. Dengan demikian, transaksi dan komitmen memastikan integritas dan konsistensi data selama operasi basis data yang kompleks.

g. ROLLBACK

```
MariaDB [rumahsakit]> START TRANSACTION;  
Query OK, 0 rows affected (0.000 sec)
```

```
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> UPDATE pembayaran SET jumlah_obat = 9, total_pembayaran = 270000, metode_pembayaran = 'Tunai'  
-> WHERE id_pembayaran = 15;  
Query OK, 1 row affected (0.002 sec)  
Rows matched: 1 Changed: 1 Warnings: 0
```

```
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> SELECT * FROM pembayaran;
```

id_pembayaran	pasien_id	periksa_kode	obat_id	jumlah_obat	total_pembayaran	metode_pembayaran	tanggal_pembayaran	pegawai_id
12	16	16	6	3	90000	Non-Tunai	2024-06-25	18
15	2	2	2	9	270000	Tunai	2024-06-19	13
21	1	1	1	2	60000	Tunai	2024-06-01	12
22	2	2	2	1	30000	Non-Tunai	2024-06-01	12
23	3	3	3	3	90000	Non-Tunai	2024-06-02	13
24	4	4	4	2	60000	Tunai	2024-06-02	13
25	5	5	5	1	30000	Non-Tunai	2024-06-03	14
26	6	6	6	3	90000	Non-Tunai	2024-06-03	14
27	7	7	7	2	60000	Tunai	2024-06-04	15
28	8	8	8	1	30000	Tunai	2024-06-04	15
29	9	9	9	3	90000	Non-Tunai	2024-06-05	16
30	10	10	10	2	60000	Tunai	2024-06-05	16
31	11	11	1	2	60000	Tunai	2024-06-06	17
32	12	12	2	1	30000	Non-Tunai	2024-06-06	17
33	13	13	3	3	90000	Non-Tunai	2024-06-07	18
34	14	14	4	2	60000	Tunai	2024-06-07	18
35	15	15	5	1	30000	Non-Tunai	2024-06-08	19
36	16	16	6	3	90000	Non-Tunai	2024-06-08	19
37	17	17	7	2	60000	Tunai	2024-06-09	20
38	18	18	8	1	30000	Non-Tunai	2024-06-09	20
39	19	19	9	3	90000	Non-Tunai	2024-06-10	12
40	20	20	10	2	60000	Tunai	2024-06-10	12

22 rows in set (0.000 sec)

```
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> ROLLBACK;  
Query OK, 0 rows affected (0.003 sec)
```

```
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> SELECT * FROM pembayaran;
```

id_pembayaran	pasien_id	periksa_kode	obat_id	jumlah_obat	total_pembayaran	metode_pembayaran	tanggal_pembayaran	pegawai_id
12	16	16	6	3	90000	Non-Tunai	2024-06-25	18
15	2	2	2	1	30000	Non-Tunai	2024-06-19	13
21	1	1	1	2	60000	Tunai	2024-06-01	12
22	2	2	2	1	30000	Non-Tunai	2024-06-01	12
23	3	3	3	3	90000	Non-Tunai	2024-06-02	13
24	4	4	4	2	60000	Tunai	2024-06-02	13
25	5	5	5	1	30000	Non-Tunai	2024-06-03	14
26	6	6	6	3	90000	Non-Tunai	2024-06-03	14
27	7	7	7	2	60000	Tunai	2024-06-04	15
28	8	8	8	1	30000	Tunai	2024-06-04	15
29	9	9	9	3	90000	Non-Tunai	2024-06-05	16
30	10	10	10	2	60000	Tunai	2024-06-05	16
31	11	11	1	2	60000	Tunai	2024-06-06	17
32	12	12	2	1	30000	Non-Tunai	2024-06-06	17
33	13	13	3	3	90000	Non-Tunai	2024-06-07	18
34	14	14	4	2	60000	Tunai	2024-06-07	18
35	15	15	5	1	30000	Non-Tunai	2024-06-08	19
36	16	16	6	3	90000	Non-Tunai	2024-06-08	19
37	17	17	7	2	60000	Tunai	2024-06-09	20
38	18	18	8	1	30000	Non-Tunai	2024-06-09	20
39	19	19	9	3	90000	Non-Tunai	2024-06-10	12
40	20	20	10	2	60000	Tunai	2024-06-10	12

22 rows in set (0.001 sec)

```
MariaDB [rumahsakit]> |
```

Rollback adalah perintah dalam basis data yang digunakan untuk membatalkan atau mengembalikan transaksi yang sedang berjalan ke keadaan sebelum transaksi dimulai. Ketika sebuah transaksi dimulai dengan perintah `'START TRANSACTION;'`, seperti yang terlihat dalam query di atas, semua perubahan data yang dilakukan selama transaksi akan disimpan secara sementara. Dalam contoh tersebut, sebuah transaksi dimulai diikuti dengan sebuah operasi UPDATE pada tabel `'pembayaran'`, yang mengubah jumlah obat, total pembayaran, dan metode pembayaran untuk entri dengan `'id_pembayaran = 15'`.

Setelah melakukan `UPDATE`, komitmen transaksi biasanya dilakukan dengan perintah `'COMMIT;'` untuk menyimpan perubahan tersebut secara permanen ke dalam basis data. Namun, jika terjadi kegagalan dalam transaksi atau ada alasan untuk membatalkan perubahan-perubahan tersebut, perintah `'ROLLBACK;'` digunakan. Dalam kasus ini, ROLLBACK digunakan untuk membatalkan perubahan yang dilakukan pada transaksi terakhir, sehingga nilai-nilai kolom untuk entri dengan `'id_pembayaran = 15'` dikembalikan ke nilai sebelum UPDATE dilakukan.

Secara keseluruhan, ROLLBACK memastikan bahwa semua perubahan yang dilakukan selama transaksi tidak disimpan secara permanen ke dalam basis data, sehingga memungkinkan untuk mengembalikan basis data ke keadaan yang konsisten sebelum transaksi dimulai jika terjadi kesalahan atau kegagalan dalam proses transaksi.

h. PROCEDURE

```
MariaDB [rumahsakit]> DELIMITER $$
MariaDB [rumahsakit]>
MariaDB [rumahsakit]> CREATE PROCEDURE TambahPegawai(
->     IN p_nama VARCHAR(255),
->     IN p_nip VARCHAR(20),
->     IN p_divisi_id INT,
->     IN p_no_telepon VARCHAR(20),
->     IN p_email VARCHAR(255),
->     IN p_tanggal_lahir DATE,
->     IN p_jenis_kelamin VARCHAR(20),
->     IN p_alamat VARCHAR(255)
-> )
-> BEGIN
->     INSERT INTO pegawai (nama, nip, divisi_id, no_telepon, email, tanggal_lahir, jenis_kelamin, alamat)
->     VALUES (p_nama, p_nip, p_divisi_id, p_no_telepon, p_email, p_tanggal_lahir, p_jenis_kelamin, p_alamat);
-> END $$
Query OK, 0 rows affected (0.008 sec)

MariaDB [rumahsakit]>
MariaDB [rumahsakit]> DELIMITER ;
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> CALL TambahPegawai(
->     'Alice Johnson',
->     '345678905',
->     1,
->     '08123456705',
->     'alicejohnson@gmail.com',
->     '1990-01-01',
->     'Perempuan',
->     'Jl. Contoh No. 1, Jakarta'
-> );
Query OK, 1 row affected (0.004 sec)

MariaDB [rumahsakit]> |
```

Procedure dalam basis data adalah sekumpulan perintah SQL yang diberi nama dan disimpan di dalam basis data untuk dieksekusi secara berulang kali. Procedure ini dapat menerima parameter sebagai input, menjalankan serangkaian perintah SQL, dan mengembalikan hasil tertentu jika diperlukan. Dalam konteks query di atas, sebuah procedure dengan nama 'TambahPegawai' dibuat untuk menambahkan data pegawai ke dalam tabel 'pegawai'.

Procedure 'TambahPegawai' menerima delapan parameter: nama pegawai, NIP, ID divisi, nomor telepon, alamat email, tanggal lahir, jenis kelamin, dan alamat. Dalam tubuh procedure, terdapat perintah INSERT yang digunakan untuk memasukkan data baru ke dalam tabel 'pegawai' menggunakan nilai-nilai yang diberikan sebagai parameter input.

Setelah procedure dibuat dengan menggunakan DELIMITER untuk mengatur penanda, panggilan procedure dilakukan dengan menggunakan perintah CALL, yang menyertakan nilai-nilai konkret untuk setiap parameter. Dalam kasus ini, panggilan procedure 'TambahPegawai' berhasil menambahkan data pegawai baru dengan nama "Alice Johnson" ke dalam tabel 'pegawai'.

i. TRIGGER

```
MariaDB [rumahsakit]> CREATE OR REPLACE TRIGGER kurangiStok
-> AFTER INSERT ON pembayaran
-> FOR EACH ROW
-> BEGIN
->   UPDATE obat
->   SET stok = stok - NEW.jumlah_obat
->   WHERE id_obat = NEW.obat_id;
-> END$$
Query OK, 0 rows affected (0.029 sec)
```

```
MariaDB [rumahsakit]> S
```

```
MariaDB [rumahsakit]> CREATE OR REPLACE TRIGGER kurangiStok
-> AFTER UPDATE ON pembayaran
-> FOR EACH ROW
-> BEGIN
-> UPDATE obat SET stok = (stok - OLD.jumlah_obat) - NEW.jumlah_obat
-> WHERE id = OLD.obat_id;
-> END$$
Query OK, 0 rows affected (0.016 sec)
```

```
MariaDB [rumahsakit]> |
```

```
MariaDB [rumahsakit]> CREATE OR REPLACE TRIGGER kembalikanStok
-> AFTER DELETE ON pembayaran
-> FOR EACH ROW
-> BEGIN
->   UPDATE obat
->   SET stok = stok + OLD.jumlah_obat
->   WHERE id_obat = OLD.obat_id;
-> END$$
Query OK, 0 rows affected (0.032 sec)
```

```
MariaDB [rumahsakit]>
```

```
MariaDB [rumahsakit]> SHOW TRIGGERS;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Trigger | Event | Table | Statement | character_set_client | collation_connection | Database Collation | Timing | Created | sql_mode |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| kurangiStok | INSERT | pembayaran | BEGIN
UPDATE obat
SET stok = stok - NEW.jumlah_obat
WHERE id_obat = NEW.obat_id;
END | AFTER | 2024-06-19 22:24:06.82 | NO_ZERO_IN_DATE,NO_ZERO_DATE,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4 | utf8mb4_general_ci | utf8_general_ci |
| updateJumlah | UPDATE | pembayaran | BEGIN
UPDATE obat SET stok = (stok + OLD.jumlah_obat) - NEW.jumlah_obat
WHERE id = OLD.obat_id;
END | AFTER | 2024-06-19 22:03:48.99 | NO_ZERO_IN_DATE,NO_ZERO_DATE,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4 | utf8mb4_general_ci | utf8_general_ci |
| kembalikanStok | DELETE | pembayaran | BEGIN
UPDATE obat
SET stok = stok + OLD.jumlah_obat
WHERE id_obat = OLD.obat_id;
END | AFTER | 2024-06-19 22:26:12.06 | NO_ZERO_IN_DATE,NO_ZERO_DATE,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4 | utf8mb4_general_ci | utf8_general_ci |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.048 sec)
MariaDB [rumahsakit]>
```

Trigger adalah objek dalam basis data yang secara otomatis dijalankan (atau dipicu) saat terjadi perubahan tertentu pada tabel, seperti perintah INSERT, UPDATE, atau DELETE. Triggers sering digunakan untuk menjaga konsistensi dan integritas data, serta untuk menjalankan otomatisasi tugas-tugas rutin tanpa intervensi manual.

Dalam konteks query yang diberikan, tiga trigger didefinisikan untuk tabel `pembayaran`:

○ Trigger `kurangiStok` (After Insert):

- Trigger ini diaktifkan setelah data baru dimasukkan ke tabel `pembayaran`.
- Setelah setiap insert, trigger ini mengurangi stok obat pada tabel `obat` berdasarkan jumlah obat yang dimasukkan pada transaksi pembayaran baru. Ini memastikan bahwa stok obat selalu up-to-date dengan penjualan yang terjadi.

○ Trigger `kurangiStok` (After Update):

- Trigger ini diaktifkan setelah data dalam tabel `pembayaran` diperbarui.
- Trigger ini memperbarui stok obat dengan mengurangi stok lama dan menambahkan stok baru berdasarkan perubahan jumlah obat yang tercatat pada transaksi pembayaran. Ini memastikan bahwa setiap perubahan jumlah obat dalam transaksi pembayaran tercermin dengan benar pada stok obat.

○ Trigger `kembalikanStok` (After Delete):

- Trigger ini diaktifkan setelah data dalam tabel `pembayaran` dihapus.
- Setelah penghapusan, trigger ini mengembalikan jumlah obat yang sesuai ke stok obat dalam tabel `obat`. Ini memastikan bahwa ketika sebuah transaksi pembayaran dihapus, stok obat yang sebelumnya terpakai dalam transaksi tersebut dikembalikan.

j. Data Controlling Language (DCL)

```
MariaDB [rumahsakit]> CREATE USER 'nurfadilah'@'0110223075' IDENTIFIED BY 'nurfadilah123';
Query OK, 0 rows affected (0.006 sec)

MariaDB [rumahsakit]>

MariaDB [rumahsakit]> GRANT ALL PRIVILEGES ON rumahsakit.* TO 'nurfadilah'@'0110223075';
Query OK, 0 rows affected (0.003 sec)

MariaDB [rumahsakit]>

MariaDB [rumahsakit]> SHOW GRANTS FOR 'nurfadilah'@'0110223075';
+-----+
| Grants for nurfadilah@0110223075 |
+-----+
| GRANT USAGE ON *.* TO 'nurfadilah'@'0110223075' IDENTIFIED BY PASSWORD '*4CB75F994F3A329C35DE7C441C53176C08BEE233' |
| GRANT ALL PRIVILEGES ON 'rumahsakit'.* TO 'nurfadilah'@'0110223075' |
| GRANT SELECT, INSERT ON 'rumahsakit'.'pegawai' TO 'nurfadilah'@'0110223075' |
| GRANT SELECT, INSERT ON 'rumahsakit'.'pembayaran' TO 'nurfadilah'@'0110223075' |
+-----+
4 rows in set (0.000 sec)

MariaDB [rumahsakit]>

MariaDB [rumahsakit]> REVOKE SELECT, INSERT ON rumahsakit.* FROM 'nurfadilah'@'0110223075';
Query OK, 0 rows affected (0.021 sec)

MariaDB [rumahsakit]>
```

Data Control Language (DCL) adalah bagian dari SQL yang digunakan untuk mengatur hak dan izin akses pengguna terhadap objek-objek dalam basis data. DCL mencakup dua perintah utama: 'GRANT' dan 'REVOKE'. Perintah 'GRANT' digunakan untuk memberikan izin kepada pengguna tertentu, sementara perintah 'REVOKE' digunakan untuk mencabut izin tersebut.

Dalam konteks query di atas, beberapa perintah DCL digunakan untuk mengelola hak akses pengguna 'nurfadilah':

- **CREATE USER:** Perintah ini membuat pengguna baru dengan nama 'nurfadilah' yang diidentifikasi dengan alamat '0110223075' dan diberi kata sandi 'nurfadilah123'. Ini adalah langkah awal dalam mengelola akses pengguna ke basis data.
- **GRANT ALL PRIVILEGES:** Perintah ini memberikan semua hak akses kepada pengguna 'nurfadilah' pada basis data 'rumahsakit'. Ini mencakup hak untuk melakukan operasi seperti SELECT, INSERT, UPDATE, DELETE, dan lainnya pada semua tabel dalam basis data tersebut.
- **SHOW GRANTS:** Perintah ini digunakan untuk menampilkan semua hak akses yang diberikan kepada pengguna 'nurfadilah'. Hasilnya menunjukkan bahwa pengguna ini memiliki berbagai hak akses termasuk ALL PRIVILEGES pada basis data 'rumahsakit', serta hak SELECT dan INSERT pada tabel 'pegawai' dan 'pembayaran'.
- **REVOKE SELECT, INSERT:** Perintah ini digunakan untuk mencabut hak akses SELECT dan INSERT dari pengguna 'nurfadilah' pada semua tabel dalam basis data 'rumahsakit'. Ini memastikan bahwa pengguna tersebut tidak lagi memiliki izin untuk melihat atau menambah data ke dalam tabel-tabel tersebut.

k. BACKUP

```
A DELL@DESKTOP-URIA4A0 c:\xampp\htdocs
# mysqldump -u root -p rumahsakit > "C:\Users\A DELL\Documents\databases\dbbackup_rumahsakit.sql"
Enter password:
```

```
A DELL@DESKTOP-URIA4A0 c:\xampp\htdocs
#
```

```
-- MariaDB dump 10.19  Distrib 10.4.32-MariaDB, for Win64 (AMD64)
--
-- Host: localhost      Database: rumahsakit
--
-- Server version      10.4.32-MariaDB

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `divisi`
--

DROP TABLE IF EXISTS `divisi`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `divisi` (
  `id_divisi` int(11) NOT NULL AUTO_INCREMENT,
  `nama_divisi` varchar(45) NOT NULL,
  PRIMARY KEY (`id_divisi`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Backup adalah proses membuat salinan data untuk melindungi informasi penting dari kehilangan atau kerusakan yang bisa terjadi akibat berbagai faktor seperti kesalahan pengguna, serangan malware, atau kerusakan perangkat keras. Dalam konteks database, backup dilakukan dengan mengekspor data ke dalam file yang bisa disimpan di lokasi aman dan digunakan untuk memulihkan data bila diperlukan.

Query di atas berkaitan dengan proses pembuatan backup database. Perintah `mysqldump -u root -p rumahsakit > "C:\Users\A DELL\Documents\databases\dbbackup_rumahsakit.sql"` menggunakan utilitas `mysqldump` untuk mengekspor semua data dari database `rumahsakit` dan menyimpannya ke dalam file `dbbackup_rumahsakit.sql` di lokasi yang ditentukan.

Perintah ini meminta pengguna untuk memasukkan password untuk akun root sebelum melanjutkan proses backup. Dengan menjalankan perintah ini, pengguna memastikan bahwa data dari database `rumahsakit` disimpan dengan aman dalam file backup, yang dapat digunakan untuk memulihkan data jika terjadi masalah.

1. RESTORE DATABASE

```
A DELL@DESKTOP-URIA4A0 c:\xampp\htdocs  
# mysql -u root -p dbrestorerumahsakit < "C:\Users\A DELL\Documents\databases\dbbackup_rumahsakit.sql"  
Enter password:
```

```
MariaDB [dbrestorerumahsakit]> SHOW TABLES;  
+-----+  
| Tables_in_dbrestorerumahsakit |  
+-----+  
| divisi  
| dokter  
| obat  
| pasien  
| pegawai  
| pembayaran  
| pendaftaran  
| pemeriksaan  
| rawat_inap  
| ruangan  
| view_pegawai_by_divisi  
| view_pegawai_detail  
| view_pegawai_perempuan  
+-----+  
13 rows in set (0.001 sec)
```

Restore adalah proses mengembalikan data dari backup ke dalam sistem yang digunakan untuk memastikan data yang hilang atau rusak dapat dipulihkan ke keadaan semula. Ini adalah langkah penting dalam manajemen data yang memastikan kontinuitas operasional dan integritas data setelah terjadi kerusakan atau kehilangan.

Dalam query di atas, serangkaian perintah dijalankan untuk melakukan restore database dari file backup. Pertama, database kosong `dbrestorerumahsakit` dibuat menggunakan perintah `CREATE DATABASE`. Kemudian, perintah `mysql -u root -p dbrestorerumahsakit < "C:\Users\A DELL\Documents\databases\dbbackup_rumahsakit.sql"` dijalankan untuk mengimpor data dari file backup `dbbackup_rumahsakit.sql` ke dalam database yang baru dibuat.

Setelah berhasil, data dalam database tersebut dapat diakses dan diverifikasi dengan menggunakan perintah `SHOW TABLES` dan `SELECT * FROM view_pegawai_by_divisi`, yang menunjukkan tabel-tabel yang telah diimpor dan data yang ada di dalamnya. Proses ini memastikan bahwa data dari database `rumahsakit` berhasil dipulihkan dan dapat digunakan kembali.