UNIVERSITAS TRUNOJOYO
MADURA

PANDUAN PRAKTIS
MEMBUAT DATABASE
STUDI KASUS DATABASE RUMAH KHITAN

- Berisi penjelasan deskripsi sistem, tampilan desain relasi antar tabel
- Tahapan pembuatan tabel data, penjelasan setiap kolom dan penjelasan trigger
- Pembuatan stored procedure dan functions
- Tampilan hasil input data

Disusun oleh:
ABDUR ROUF

Dosen pembimbing:
MOCH KAUTSAR SOPHAN, S.Kom., M.MT.

2020
SISTEM INFORMASI

PANDUAN PRAKTIS MEMBUAT DATABASE
STUDI KASUS DATABASE RUMAH KHITAN

Disusun untuk Memenuhi Tugas Matakuliah Basis Data Lanjut A
Program Studi Sistem Informasi Jenjang Sarjana

Disusun oleh:
Abdur Rouf
NIM. 170441100037

Dosen Pembimbing:
Moch. Kautsar Sophan, S.Kom. M.MT.

PROGRAM STUDI S1 SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA
2020

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas rahmat dan karunia-Nya, sehingga buku panduan yang berjudul “Panduan Praktis Membuat Database Studi Kasus Database Rumah Khitan” ini dapat terselesaikan dengan baik dan tepat pada waktunya. Tujuan dari pembuatan buku panduan ini adalah untuk memberikan informasi dan wawasan tambahan kepada pembaca mengenai bagaimana membuat desain database dengan baik, serta dapat dipahami oleh sesama pengelola basis data khususnya pada database rumah khitan. Dan, pembaca dapat juga mempraktikan dan mengimplementasikan dasar-dasar pembuatan database pada studi kasus yang lain.

Buku Panduan ini dapat terselesaikan dengan baik atas saran dan bimbingan Bapak Moch Kautsar Sophan, S.Kom., M.MT. selaku pengampu mata kuliah Basis Data Lanjut. Akhirnya, penulis sampaikan terima kasih kepada semua pihak yang telah membantu penulis dalam mewujudkan Buku Panduan ini. Semoga buku panduan ini dapat bermanfaat untuk pembaca.

Bangkalan, 15 Juni 2020
Penulis,

ABDUR ROUF

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

A. Deskripsi Sistem

Rumah Khitan merupakan usaha jasa di bidang kesehatan terutama pada layanan khitan bagi masyarakat. Khitan adalah suatu prosedur medis yang paling sering dilakukan di dunia kesehatan. Kegiatan khitan sudah menjadi kebudayaan bangsa Indonesia karena selain perintah agama (islam), khitan menjadi salah satu tindakan untuk menjaga kesehatan.

Rumah khitan menjadi salah satu usaha yang banyak dijalankan di ranah medis. Dalam menjalankan usaha tersebut, rumah khitan umumnya masih menggunakan pengelolaan data transaksinya berbasis file *hardcopy*. Keadaan tersebut tentu akan muncul masalah seperti sulitnya pencarian data jika datanya terlampau banyak, dan duplikasi data tidak terkontrol dengan baik. Akibatnya, pelayanan rumah khitan akan mengalami penurunan.

Untuk mengatasi masalah tersebut, perlu adanya perancangan suatu database untuk mempermudah pencatatan dan pengelolaan data transaksionalnya agar dapat meningkatkan pelayanan rumah khitan. Penggunaan dari sistem basis data rumah khitan nantinya akan mampu menyimpan dan mengontrol data pasien serta transaksi atau layanan apa saja yang telah dilakukan.

Database Rumah Khitan ini terdapat 4 layanan yaitu layanan khitan (sebagai layanan utama), layanan konsultasi, layanan beli obat, dan layanan beli alat khitan. Dalam layanan khitan terdapat beberapa metode yang ditawarkan dengan jumlah biaya yang berbeda-beda. Biaya layanan akan dihitung otomatis dalam sistem database. Selain pengelolaan layanan, sistem database ini akan mengelola pengadaan alat dan obat untuk menunjang jalannya layanan rumah khitan. Dengan demikian, pengguna dan penyedia layanan rumah khitan akan mendapat kemudahan dari sistem database ini.

B. Ruang Lingkup dan Tujuan

Ruang lingkup dari pembuatan Database Rumah Khitan ini yaitu

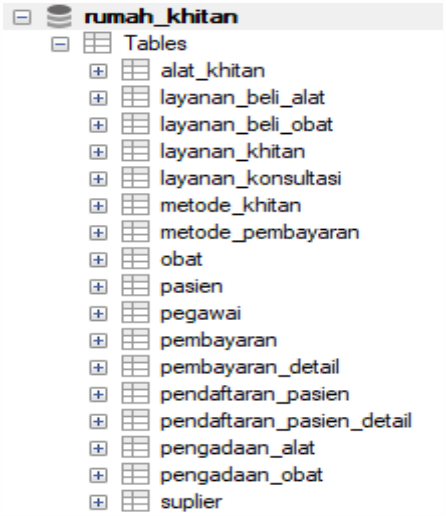
1. Tabel layanan Rumah Khitan seperti yang telah disebutkan di deskripsi sistem.

2. Database melakukan perhitungan biaya di masing-masing layanan dan total pembayaran.
3. Database melakukan perhitungan persediaan peralatan khitan/ persediaan obat khitan.
4. Database melakukan perhitungan total pengeluaran dan pemasukan.

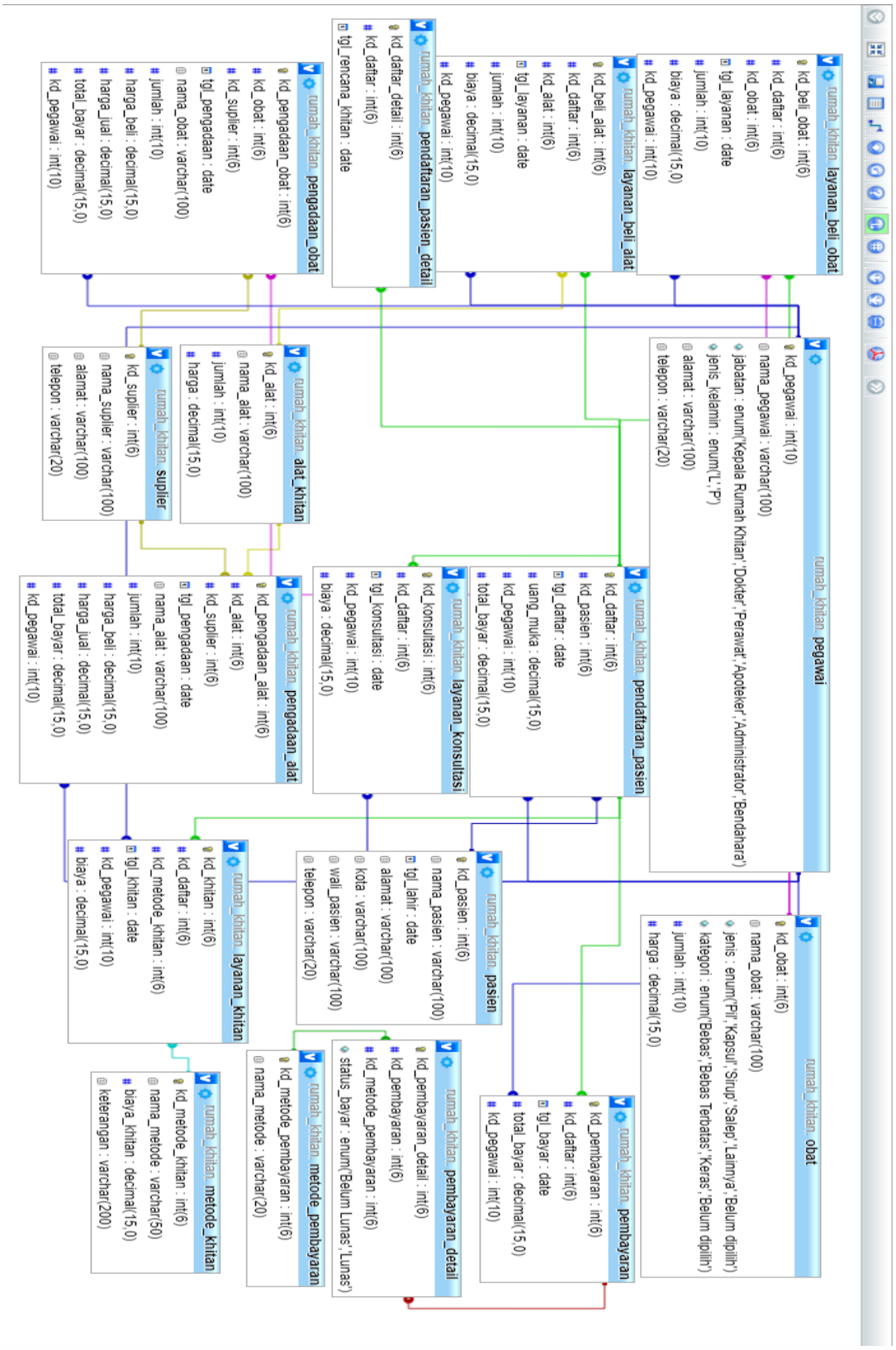
Tujuan pembuatan database ini untuk menganalisis dan mengolah data transaksi rumah khitan dalam sistem database agar natinya dapat dilaporkan sewaktu-waktu dengan baik.

C. Desain Relasi Antar Tabel

Tabel Data yang digunakan dalam Database Rumah Khitan terdiri dari Tabel Master dan Tabel Transaksi. Tabel Master dalam database rumah khitan meliputi Tabel Pasien, Tabel Pegawai, Tabel Metode Khitan, Tabel Alat Khitan, Tabel Obat Khitan, Tabel Suplier, dan Tabel Metode Pembayaran. Tabel Transaksi meliputi Tabel Pendaftaran Pasien, Tabel Pendaftaran Pasien Detail, Tabel Layanan Khitan, Tabel Layanan Konsultasi, Tabel Layanan Beli Alat Khitan, Tabel Layanan Beli Obat Khitan, Tabel Pengadaan Alat Khitan, Tabel Pengadaan Obat Khitan, Tabel Pembayaran dan Pembayaran Detail. Desain relasi antar tabel tergambarkan dalam (Gambar 2).



Gambar 1. Tabel Data dalam database Rumah Khitan



Gambar 2. Desain Relasi Antar Tabel

BAB II
PEMBUATAN TABEL DATA DAN *TRIGGER*

Sebelum mulai membuat tabel data dan *trigger*, dapat diawali dengan membuat database yaitu dengan menuliskan Query SQL:

```
CREATE DATABASE rumah_khitan;
```

A. Pembuatan Tabel Master

Tabel Master adalah tabel independen yang datanya cenderung tidak berubah-ubah. Data tabel master berguna untuk memberikan informasi tambahan mengenai proses pengelolaan data berikutnya (dalam tabel transaksi). Berikut ini akan dijelaskan mengenai pembuatan tabel master database rumah khitan:

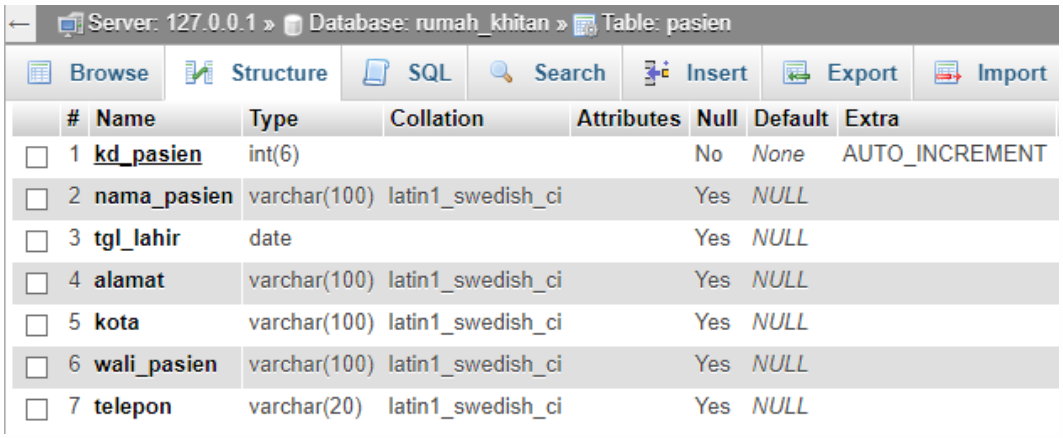
1. Membuat Tabel Pasien

   Query SQL:

```
CREATE TABLE `pasien` (
  `kd_pasien` int(6) NOT NULL AUTO_INCREMENT,
  `nama_pasien` varchar(100) DEFAULT NULL,
  `tgl_lahir` date DEFAULT NULL,
  `alamat` varchar(100) DEFAULT NULL,
  `kota` varchar(100) DEFAULT NULL,
  `wali_pasien` varchar(100) DEFAULT NULL,
  `telepon` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`kd_pasien`)
);
```

Gambar 3. Query Pembuatan Tabel Pasien

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_pasien	int(6)			No	None	AUTO_INCREMENT
2	nama_pasien	varchar(100)	latin1_swedish_ci		Yes	NULL	
3	tgl_lahir	date			Yes	NULL	
4	alamat	varchar(100)	latin1_swedish_ci		Yes	NULL	
5	kota	varchar(100)	latin1_swedish_ci		Yes	NULL	
6	wali_pasien	varchar(100)	latin1_swedish_ci		Yes	NULL	
7	telepon	varchar(20)	latin1_swedish_ci		Yes	NULL	

Gambar 4. Struktur Tabel Pasien

Penjelasan:

Pembuatan tabel di atas bernama ‘pasien’, terdiri dari 7 atribut data. Atribut ‘kd_pasien’ bertipe data **integer** dan **auto_increment** artinya kd_pasien akan bertambah secara otomatis nilainya jika terjadi penambahan *row* pada table dimana field tersebut berada. Kemudian ada atribut ‘nama_pasien’ pendaftar layanan khitan bertipe data varchar (100). Atribut tgl_lahir bertipe data DATE untuk menyimpan data kelahiran pasien. Atribut ‘alamat’ bertipe data varchar (100) untuk menyimpan alamat. Atribut ‘kota’ bertipe data varchar (100). Atribut ‘wali_pasien’ bertipe data varchar (100) untuk menyimpan data nama orang tua atau pendamping pasien khitan. Dan Atribut telepon bertipe data varchar (20). Data default dalam tabel pasien dapat bernilai **NULL** kecuali pada atribut ‘kd_pasien’ karena **auto_increment** dan tidak mungkin kosong datanya. Atribut ‘kd_pasien’ berlaku sebagai kode unik pada tabel pasien.

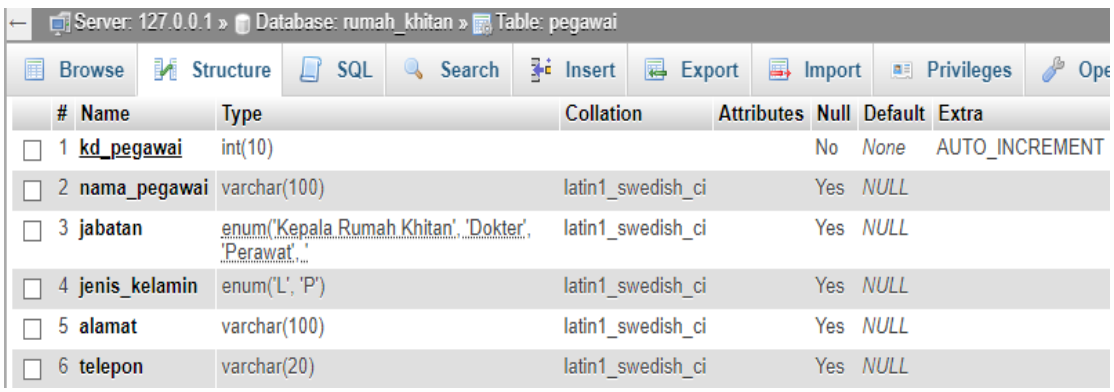
2. Membuat Tabel Pegawai

Query SQL:

```
CREATE TABLE `pegawai` (
  `kd_pegawai` int(10) NOT NULL AUTO_INCREMENT,
  `nama_pegawai` varchar(100) DEFAULT NULL,
  `jabatan` enum('Kepala Rumah Khitan','Dokter','Perawat',
  'Apoteker','Administrator','Bendahara') DEFAULT NULL,
  `jenis_kelamin` enum('L','P') DEFAULT NULL,
  `alamat` varchar(100) DEFAULT NULL,
  `telepon` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`kd_pegawai`)
);
```

Gambar 5. Query Pembuatan Tabel Pegawai

Hasil:



Gambar 6. Struktur Tabel Pegawai

Penjelasan:

Pembuatan tabel di atas bernama ‘pegawai’, terdiri dari 6 atribut data yaitu `kd_pegawai` bertipe data integer dengan panjang maksimal (10 karakter) yang datanya tidak boleh kosong dan **auto_increment** (bertambah secara otomatis nilainya jika terjadi penambahan *row*), ada atribut `nama_pegawai` bertipe data varchar(100), atribut `jabatan` bertipe data enum dengan pilihan ('Kepala Rumah Khitan', 'Dokter', 'Perawat', 'Apoteker', 'Administrator', 'Bendahara'), ada atribut `jenis_kelamin` dengan tipe data enum dan pilihan data ('L' dan 'P'), ada atribut `alamat` bertipe data varchar(100), dan atribut `telepon` bertipe data varchar(20). Data default dalam tabel pegawai dapat bernilai **NULL** kecuali pada atribut kd_pegawai. Atribut ‘kd_pegawai’ juga berlaku sebagai kode unik pada tabel pegawai.

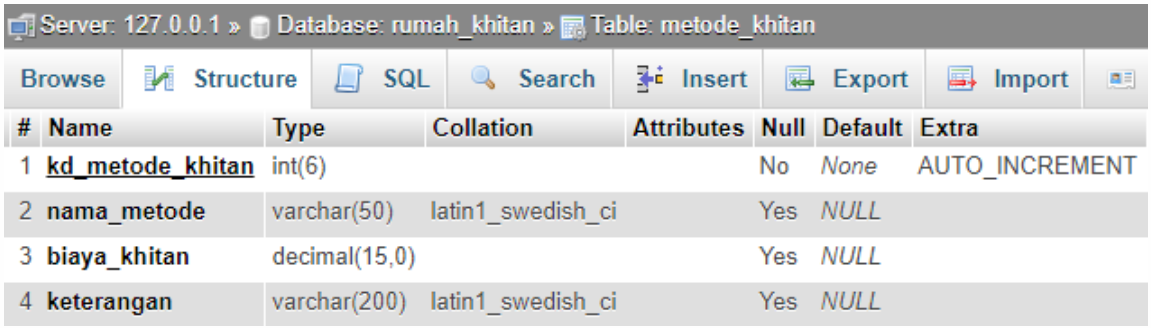
3. Membuat Tabel Metode Khitan

Query SQL:

```
CREATE TABLE `metode_khitan` (
  `kd_metode_khitan` int(6) NOT NULL AUTO_INCREMENT,
  `nama_metode` varchar(50) DEFAULT NULL,
  `biaya_khitan` decimal(15,0) DEFAULT NULL,
  `keterangan` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`kd_metode_khitan`)
);
```

Gambar 7. Query Pembuatan Tabel Metode Khitan

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_metode_khitan	int(6)			No	None	AUTO_INCREMENT
2	nama_metode	varchar(50)	latin1_swedish_ci		Yes	NULL	
3	biaya_khitan	decimal(15,0)			Yes	NULL	
4	keterangan	varchar(200)	latin1_swedish_ci		Yes	NULL	

Gambar 8. Struktur Tabel Metode Khitan

Penjelasan:

Pembuatan tabel di atas bernama ‘metode_khitan’, terdiri dari 4 atribut data yaitu `kd_metode_khitan` bertipe data int(6) dan **auto_increment** (bertambah

secara otomatis nilainya jika terjadi penambahan *row*), atribut `nama_metode` bertipe data varchar(50), atribut `biaya_khitan` bertipe data decimal(15,0) dan akan digunakan dalam perhitungan transaksional layanan khitan nantinya, atribut `keterangan` bertipe data varchar(200) sebagai penjelasan dari tiap-tiap 'nama_metode' khitan. Data default dalam tabel 'metode_khitan' dapat bernilai **NULL** kecuali pada atribut kd_metode_khitan. Atribut kd_metode_khitan juga berlaku sebagai kode unik pada tabel 'metode_khitan'.

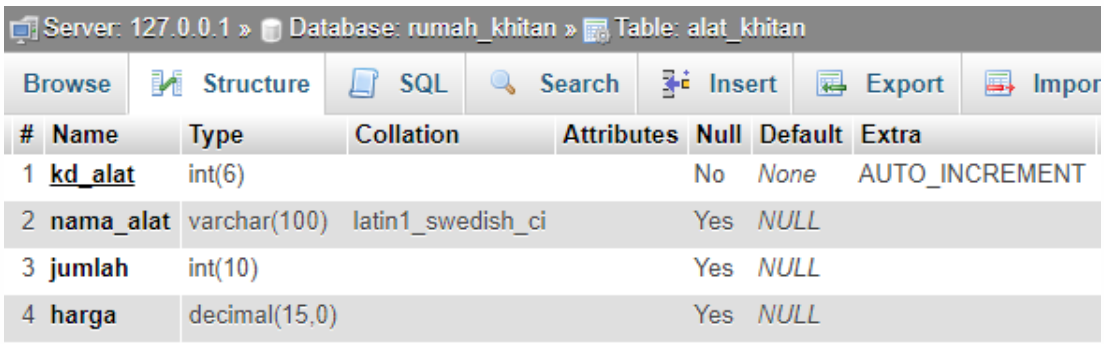
4. Membuat Tabel Alat Khitan

Query SQL:

```
CREATE TABLE `alat_khitan` (
  `kd_alat` int(6) NOT NULL AUTO_INCREMENT,
  `nama_alat` varchar(100) DEFAULT NULL,
  `jumlah` int(10) DEFAULT NULL,
  `harga` decimal(15,0) DEFAULT NULL,
  PRIMARY KEY (`kd_alat`)
);
```

Gambar 9. Query Pembuatan Tabel Alat Khitan

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_alat	int(6)			No	*None*	AUTO_INCREMENT
2	nama_alat	varchar(100)	latin1_swedish_ci		Yes	*NULL*	
3	jumlah	int(10)			Yes	*NULL*	
4	harga	decimal(15,0)			Yes	*NULL*	

Gambar 10. Struktur Tabel Alat Khitan

Penjelasan:

Pembuatan tabel di atas bernama `alat_khitan`, terdiri dari 4 atribut data yaitu atribut `kd_alat` bertipe data int(6) dan **auto_increment**, atribut `nama_alat` bertipe data varchar(100), atribut `jumlah` bertipe data int(10), atribut `harga` bertipe data decimal(15,0) untuk perhitungan layanan beli alat nantinya. Data default dalam tabel `alat_khitan` dapat bernilai **NULL** kecuali pada atribut `kd_alat`. Atribut `kd_alat` juga berlaku sebagai kode unik pada tabel `alat_khitan`.

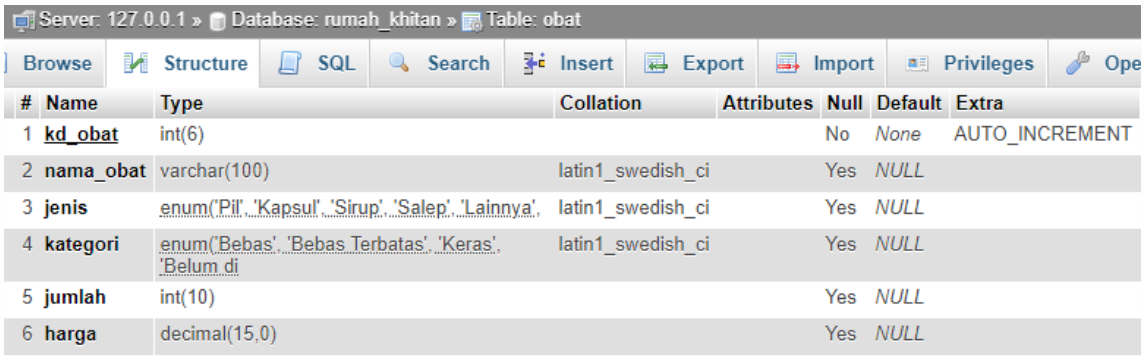
5. Membuat Tabel Obat Khitan

Query SQL:

```
CREATE TABLE `obat` (
  `kd_obat` int(6) NOT NULL AUTO_INCREMENT,
  `nama_obat` varchar(100) DEFAULT NULL,
  `jenis` enum('Pil','Kapsul','Sirup','Salep','Lainnya','Belum
  dipilih') DEFAULT NULL,
  `kategori` enum('Bebas','Bebas Terbatas','Keras','Belum
  dipilih') DEFAULT NULL,
  `jumlah` int(10) DEFAULT NULL,
  `harga` decimal(15,0) DEFAULT NULL,
  PRIMARY KEY (`kd_obat`)
);
```

Gambar 11. Query Pembuatan Tabel Obat Khitan

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_obat	int(6)			No	None	AUTO_INCREMENT
2	nama_obat	varchar(100)	latin1_swedish_ci		Yes	NULL	
3	jenis	enum('Pil', 'Kapsul', 'Sirup', 'Salep', 'Lainnya',	latin1_swedish_ci		Yes	NULL	
4	kategori	enum('Bebas', 'Bebas Terbatas', 'Keras', 'Belum di	latin1_swedish_ci		Yes	NULL	
5	jumlah	int(10)			Yes	NULL	
6	harga	decimal(15,0)			Yes	NULL	

Gambar 12. Struktur Tabel Obat Khitan

Penjelasan:

Pembuatan tabel di atas bernama ‘obat’, terdiri dari 6 atribut data yaitu atribut `kd_obat` yang bertipe data int(6) dan **auto_increment**, atribut `nama_obat` bertipe data varchar(100), atribut `jenis` bertipe data enum dengan pilihan data ('Pil', 'Kapsul', 'Sirup', 'Salep', 'Lainnya', dan 'Belum dipilih'), atribut `kategori` bertipe data enum ('Bebas', 'Bebas Terbatas', 'Keras', dan 'Belum dipilih'), atribut `jumlah` bertipe data int(10), dan atribut `harga` bertipe data decimal(15,0) untuk perhitungan layanan beli obat nantinya. Data default dalam tabel ‘obat’ dapat bernilai **NULL** kecuali pada atribut `kd_obat`. Atribut `kd_obat` juga berlaku sebagai kode unik pada tabel ‘obat’.

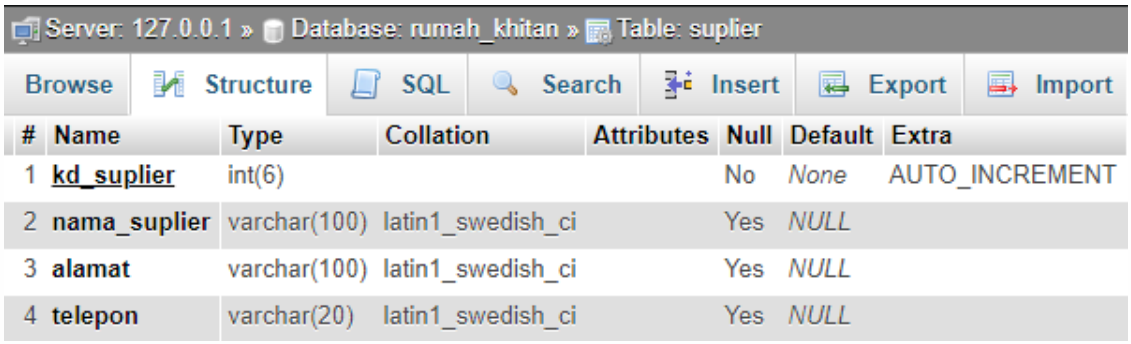
6. Membuat Tabel Suplier

Query SQL:

```
CREATE TABLE `suplier` (
  `kd_suplier` int(6) NOT NULL AUTO_INCREMENT,
  `nama_suplier` varchar(100) DEFAULT NULL,
  `alamat` varchar(100) DEFAULT NULL,
  `telepon` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`kd_suplier`)
);
```

Gambar 13. Query Pembuatan Tabel Suplier

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_suplier	int(6)			No	None	AUTO_INCREMENT
2	nama_suplier	varchar(100)	latin1_swedish_ci		Yes	NULL	
3	alamat	varchar(100)	latin1_swedish_ci		Yes	NULL	
4	telepon	varchar(20)	latin1_swedish_ci		Yes	NULL	

Gambar 14. Struktur Tabel Suplier

Penjelasan:

Pembuatan tabel di atas bernama `suplier`, terdiri dari 4 atribut data yaitu atribut `kd_suplier` bertipe data int(6) dan **auto_increment**, atribut `nama_suplier` bertipe data varchar(100), atribut `alamat` bertipe data varchar(100, dan atribut `telepon` bertipe data varchar(20). Data default dalam tabel `suplier` dapat bernilai **NULL** kecuali pada atribut `kd_suplier`. Atribut `kd_suplier` juga berlaku sebagai kode unik pada tabel `suplier`.

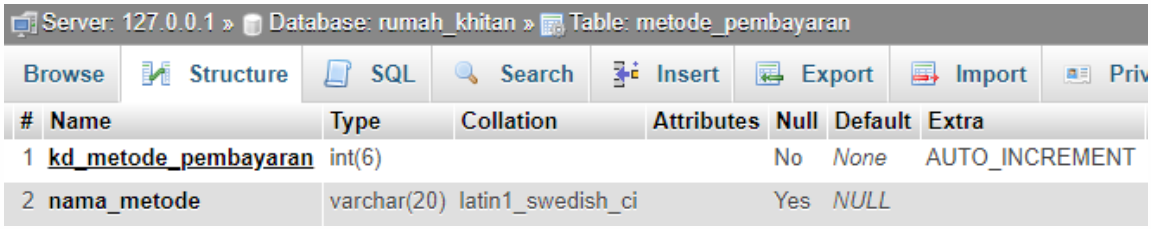
7. Membuat Tabel Metode Pembayaran

Query SQL:

```
CREATE TABLE `metode_pembayaran` (
  `kd_metode_pembayaran` int(6) NOT NULL AUTO_INCREMENT,
  `nama_metode` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`kd_metode_pembayaran`)
);
```

Gambar 15. Query Pembuatan Tabel Metode Pembayaran

Hasil:



Gambar 16. Struktur Tabel Metode Pembayaran

Penjelasan:

Pembuatan tabel di atas bernama `metode_pembayaran`, terdiri dari 2 atribut data yaitu atribut `kd_metode_pembayaran` bertipe data int (6) dan **auto_increment**, dan atribut `nama_metode` bertipe data varchar (20). Data default dalam tabel `metode_pembayaran` dapat bernilai **NULL** kecuali pada atribut `kd_metode_pembayaran`. Atribut `kd_metode_pembayaran` juga berlaku sebagai kode unik pada tabel `metode_pembayaran`.

B. Pembuatan Tabel Transaksi

Tabel Transaksi adalah tabel yang bersifat relatif atau isi datanya dapat berubah-ubah. Tabel transaksi tidak dapat berdiri sendiri tanpa adanya tabel master atau dapat juga bergantung dengan tabel transaksi lainnya. Ketergantungan ini dihubungkan dengan adanya relasi antar tabel. Berikut ini akan dijelaskan mengenai pembuatan tabel transaksi database rumah khitan:

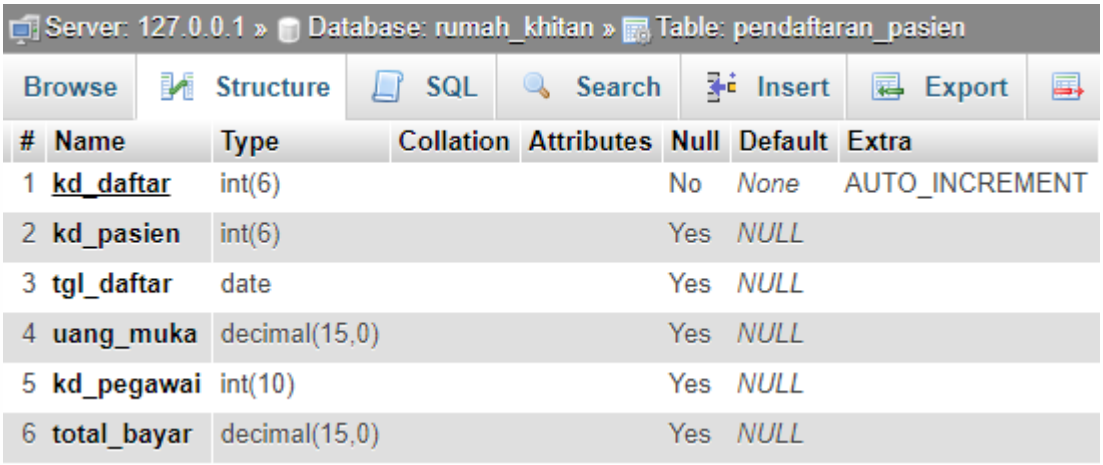
1. Membuat Tabel Pendaftaran Pasien

   Query SQL:

```
CREATE TABLE `pendaftaran_pasien` (
  `kd_daftar` int(6) NOT NULL AUTO_INCREMENT,
  `kd_pasien` int(6) DEFAULT NULL,
  `tgl_daftar` date DEFAULT NULL,
  `uang_muka` decimal(15,0) DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  `total_bayar` decimal(15,0) DEFAULT NULL,
  PRIMARY KEY (`kd_daftar`),
  KEY `kd_pasien` (`kd_pasien`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `pendaftaran_pasien_ibfk_1` FOREIGN KEY (`kd_pasien
  `) REFERENCES `pasien` (`kd_pasien`),
  CONSTRAINT `pendaftaran_pasien_ibfk_2` FOREIGN KEY (`
  kd_pegawai`) REFERENCES `pegawai` (`kd_pegawai`)
);
```

Gambar 17. Query Pembuatan Tabel Pendaftaran Pasien

Hasil:


Server: 127.0.0.1 » Database: rumah_khitan » Table: pendaftaran_pasien

Browse | Structure | SQL | Search | Insert | Export

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_daftar	int(6)			No	None	AUTO_INCREMENT
2	kd_pasien	int(6)			Yes	NULL	
3	tgl_daftar	date			Yes	NULL	
4	uang_muka	decimal(15,0)			Yes	NULL	
5	kd_pegawai	int(10)			Yes	NULL	
6	total_bayar	decimal(15,0)			Yes	NULL	

Gambar 18. Struktur Tabel Pendaftaran Pasien

Penjelasan:

Pembuatan tabel di atas bernama `pendaftaran_pasien`, terdiri dari 6 atribut data yaitu atribut `kd_daftar` bertipe data int(6) dan **auto_increment,** atribut `kd_pasien` bertipe data int(6), atribut `tgl_daftar` bertipe data, atribut `uang_muka` bertipe data decimal, atribut `kd_pegawai` bertipe data int(10), atribut `total_bayar` bertipe data decimal(15,0). Data default dalam tabel `pendaftaran_pasien` dapat bernilai **NULL** kecuali pada atribut `kd_daftar`. Atribut `kd_daftar` juga berlaku sebagai kode unik pada tabel `pendaftaran_pasien`. Atribut `kd_pasien` berlaku sebagai kode referensi (untuk membuat relasi tabel) dari tabel 'pasien'. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel 'pegawai'.

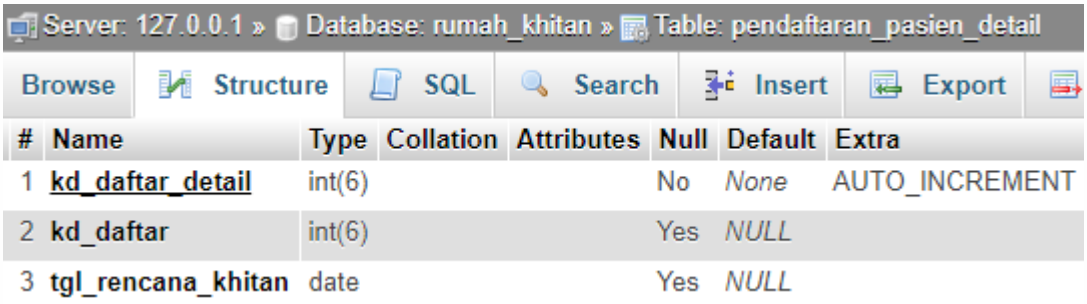
2. Membuat Tabel Pendaftaran Pasien Detail

Query SQL:

```
CREATE TABLE `pendaftaran_pasien_detail` (
  `kd_daftar_detail` int(6) NOT NULL AUTO_INCREMENT,
  `kd_daftar` int(6) DEFAULT NULL,
  `tgl_rencana_khitan` date DEFAULT NULL,
  PRIMARY KEY (`kd_daftar_detail`),
  KEY `kd_daftar` (`kd_daftar`),
  CONSTRAINT `pendaftaran_pasien_detail_ibfk_1` FOREIGN KEY (`
  kd_daftar`) REFERENCES `pendaftaran_pasien` (`kd_daftar`)
);
```

Gambar 19. Query Pembuatan Tabel Pendaftaran Pasien Detail

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_daftar_detail	int(6)			No	None	AUTO_INCREMENT
2	kd_daftar	int(6)			Yes	NULL	
3	tgl_rencana_khitan	date			Yes	NULL	

Gambar 20. Struktur Tabel Pendaftaran Pasien Detail

Penjelasan:

Pembuatan tabel di atas bernama `pendaftaran_pasien_detail`, terdiri dari 3 atribut data yaitu atribut `kd_daftar_detail` bertipe data int(6) dan **auto_increment,** atribut bertipe data `kd_daftar` int(6), dan atribut `tgl_rencana_khitan` bertipe data date. Data default dalam tabel `pendaftaran_pasien_detail` dapat bernilai **NULL** kecuali pada atribut `kd_daftar_detail`. Atribut `kd_daftar_detail` juga berlaku sebagai kode unik pada tabel `pendaftaran_pasien_detail`. Atribut `kd_daftar` berlaku sebagai kode yang bereferensi dari tabel `pendaftaran_pasien`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel 'pegawai'.

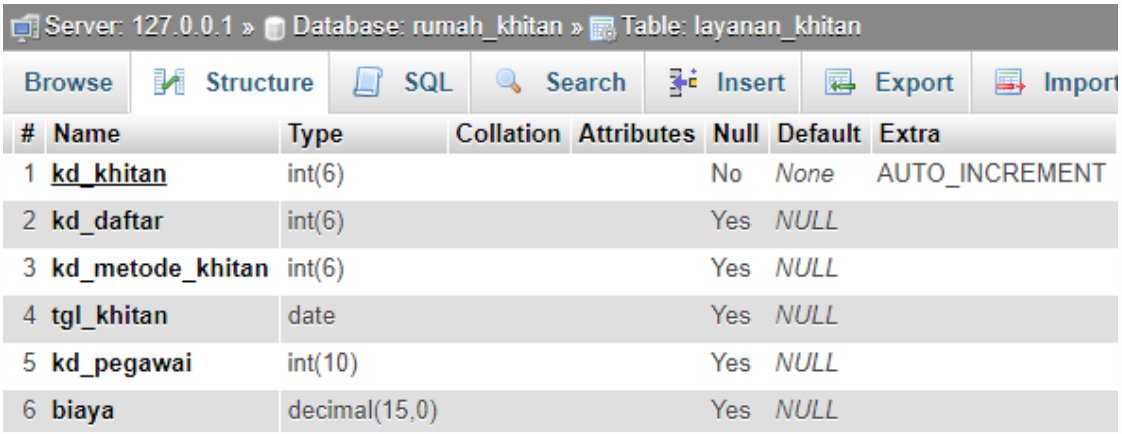
3. Membuat Tabel Layanan Khitan

Query SQL:

```
CREATE TABLE `layanan_khitan` (
  `kd_khitan` int(6) NOT NULL AUTO_INCREMENT,
  `kd_daftar` int(6) DEFAULT NULL,
  `kd_metode_khitan` int(6) DEFAULT NULL,
  `tgl_khitan` date DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  `biaya` decimal(15,0) DEFAULT NULL,
  PRIMARY KEY (`kd_khitan`),
  KEY `kd_daftar` (`kd_daftar`),
  KEY `kd_metode_khitan` (`kd_metode_khitan`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `layanan_khitan_ibfk_1` FOREIGN KEY (`kd_daftar`)
  REFERENCES `pendaftaran_pasien` (`kd_daftar`),
  CONSTRAINT `layanan_khitan_ibfk_2` FOREIGN KEY (`kd_pegawai`)
  REFERENCES `pegawai` (`kd_pegawai`),
  CONSTRAINT `layanan_khitan_ibfk_3` FOREIGN KEY (`
  kd_metode_khitan`) REFERENCES `metode_khitan` (`
  kd_metode_khitan`)
);
```

Gambar 21. Query Pembuatan Tabel Layanan Khitan

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_khitan	int(6)			No	None	AUTO_INCREMENT
2	kd_daftar	int(6)			Yes	NULL	
3	kd_metode_khitan	int(6)			Yes	NULL	
4	tgl_khitan	date			Yes	NULL	
5	kd_pegawai	int(10)			Yes	NULL	
6	biaya	decimal(15,0)			Yes	NULL	

Gambar 22. Struktur Tabel Layanan Khitan

Penjelasan:

Pembuatan tabel di atas bernama `layanan_khitan`, terdiri dari 6 atribut data yaitu atribut `kd_khitan` bertipe data int(6) dan **auto_increment**. atribut, atribut `kd_daftar` bertipe data int(6), `kd_metode_khitan` bertipe data int(6), `tgl_khitan` bertipe data date, atribut `kd_pegawai` bertipe data int(10) dan `biaya` bertipe data decimal(15,0). Data default dalam tabel `layanan_khitan` dapat bernilai **NULL** kecuali pada atribut `kd_khitan`. Atribut `kd_khitan` juga berlaku sebagai kode unik pada tabel `layanan_khitan`. Atribut `kd_daftar` berlaku sebagai kode bereferensi dari tabel `pendaftaran_pasien`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel 'pegawai'.

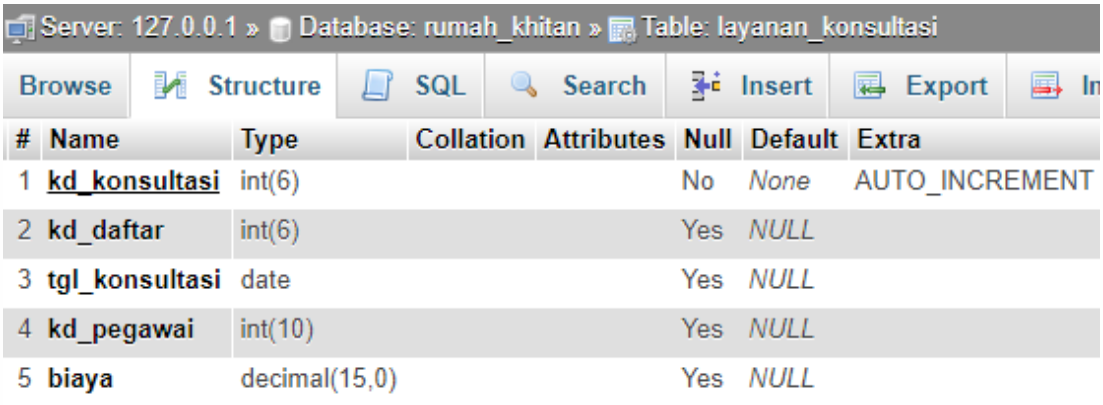
4. Membuat Tabel Layanan Konsultasi

Query SQL:

```
CREATE TABLE `layanan_konsultasi` (
  `kd_konsultasi` int(6) NOT NULL AUTO_INCREMENT,
  `kd_daftar` int(6) DEFAULT NULL,
  `tgl_konsultasi` date DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  `biaya` decimal(15,0) DEFAULT NULL,
  PRIMARY KEY (`kd_konsultasi`),
  KEY `kd_daftar` (`kd_daftar`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `layanan_konsultasi_ibfk_1` FOREIGN KEY (`kd_daftar
  `) REFERENCES `pendaftaran_pasien` (`kd_daftar`),
  CONSTRAINT `layanan_konsultasi_ibfk_2` FOREIGN KEY (`
  kd_pegawai`) REFERENCES `pegawai` (`kd_pegawai`)
);
```

Gambar 23. Query Pembuatan Tabel Layanan Konsultasi

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_konsultasi	int(6)			No	None	AUTO_INCREMENT
2	kd_daftar	int(6)			Yes	NULL	
3	tgl_konsultasi	date			Yes	NULL	
4	kd_pegawai	int(10)			Yes	NULL	
5	biaya	decimal(15,0)			Yes	NULL	

Gambar 24. Struktur Tabel Layanan Konsultasi

Penjelasan:

Pembuatan tabel di atas bernama `layanan_konsultasi`, terdiri dari 5 atribut data yaitu atribut `kd_konsultasi` bertipe data int(6) dan **auto_increment,** atribut `kd_daftar` bertipe data int(6), atribut `tgl_konsultasi` bertipe data date, atribut `kd_pegawai` bertipe data int(10), dan atribut `biaya` bertipe data decimal(15,0). Data default dalam tabel `layanan_konsultasi` dapat bernilai **NULL** kecuali pada atribut `kd_konsultasi`. Atribut `kd_konsultasi` juga berlaku sebagai kode unik pada tabel `layanan_konsultasi`. Atribut `kd_daftar` berlaku sebagai kode yang bereferensi dari tabel `pendaftaran_pasien`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel ‘pegawai’.

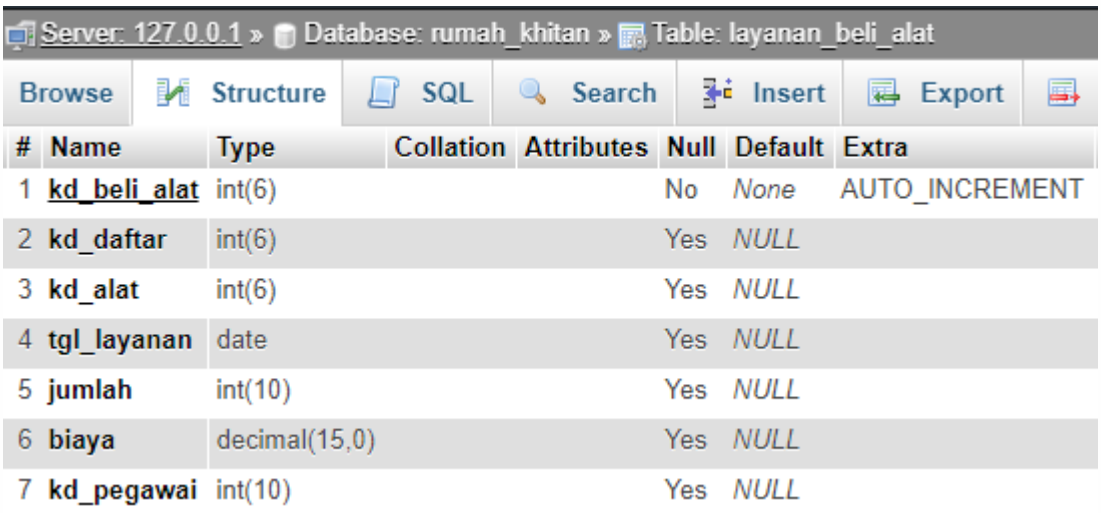
5. Membuat Tabel Layanan Beli Alat Khitan

Query SQL:

```
CREATE TABLE `layanan_beli_alat` (
  `kd_beli_alat` int(6) NOT NULL AUTO_INCREMENT,
  `kd_daftar` int(6) DEFAULT NULL,
  `kd_alat` int(6) DEFAULT NULL,
  `tgl_layanan` date DEFAULT NULL,
  `jumlah` int(10) DEFAULT NULL,
  `biaya` decimal(15,0) DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  PRIMARY KEY (`kd_beli_alat`),
  KEY `kd_daftar` (`kd_daftar`),
  KEY `kd_alat` (`kd_alat`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `layanan_beli_alat_ibfk_1` FOREIGN KEY (`kd_daftar`
  ) REFERENCES `pendaftaran_pasien` (`kd_daftar`),
  CONSTRAINT `layanan_beli_alat_ibfk_2` FOREIGN KEY (`kd_pegawai
  `) REFERENCES `pegawai` (`kd_pegawai`),
  CONSTRAINT `layanan_beli_alat_ibfk_3` FOREIGN KEY (`kd_alat`)
  REFERENCES `alat_khitan` (`kd_alat`)
);
```

Gambar 25. Query Pembuatan Tabel Layanan Beli Alat Khitan

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_beli_alat	int(6)			No	None	AUTO_INCREMENT
2	kd_daftar	int(6)			Yes	NULL	
3	kd_alat	int(6)			Yes	NULL	
4	tgl_layanan	date			Yes	NULL	
5	jumlah	int(10)			Yes	NULL	
6	biaya	decimal(15,0)			Yes	NULL	
7	kd_pegawai	int(10)			Yes	NULL	

Gambar 26. Struktur Tabel Layanan Beli Alat Khitan

Penjelasan:

Pembuatan tabel di atas bernama `layanan_beli_alat`, terdiri dari 7 atribut data yaitu atribut `kd_beli_alat` bertipe data int(6) dan **auto_increment,** atribut `kd_daftar` bertipe data int(6), `kd_alat` bertipe data int(6), atribut `tgl_layanan` bertipe data, atribut `jumlah` bertipe data int(10), atribut `biaya` bertipe data decimal(15,0), dan atribut `kd_pegawai` bertipe data int(10). Data default dalam tabel `layanan_beli_alat` dapat bernilai **NULL** kecuali pada atribut `kd_beli_alat`.

Atribut `kd_beli_alat` juga berlaku sebagai kode unik pada tabel `layanan_beli_alat`. Atribut `kd_daftar` berlaku sebagai kode yang bereferensi dari tabel `pendaftaran_pasien`. Atribut `kd_alat` berlaku sebagai kode yang bereferensi dari tabel 'alat_khitan`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel 'pegawai'.

6. Membuat Tabel Layanan Beli Obat Khitan

Query SQL:

```
CREATE TABLE `layanan_beli_obat` (
  `kd_beli_obat` int(6) NOT NULL AUTO_INCREMENT,
  `kd_daftar` int(6) DEFAULT NULL,
  `kd_obat` int(6) DEFAULT NULL,
  `tgl_layanan` date DEFAULT NULL,
  `jumlah` int(10) DEFAULT NULL,
  `biaya` decimal(15,0) DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  PRIMARY KEY (`kd_beli_obat`),
  KEY `kd_daftar` (`kd_daftar`),
  KEY `kd_obat` (`kd_obat`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `layanan_beli_obat_ibfk_1` FOREIGN KEY (`kd_daftar`
  ) REFERENCES `pendaftaran_pasien` (`kd_daftar`),
  CONSTRAINT `layanan_beli_obat_ibfk_2` FOREIGN KEY (`kd_pegawai
  `) REFERENCES `pegawai` (`kd_pegawai`),
  CONSTRAINT `layanan_beli_obat_ibfk_3` FOREIGN KEY (`kd_obat`)
  REFERENCES `obat` (`kd_obat`)
);
```

Gambar 27. Query Pembuatan Tabel Layanan Beli Obat Khitan

Hasil:

Server: 127.0.0.1 » Database: rumah_khitan » Table: layanan_beli_obat

Browse | Structure | SQL | Search | Insert | Export

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_beli_obat	int(6)			No	None	AUTO_INCREMENT
2	kd_daftar	int(6)			Yes	NULL	
3	kd_obat	int(6)			Yes	NULL	
4	tgl_layanan	date			Yes	NULL	
5	jumlah	int(10)			Yes	NULL	
6	biaya	decimal(15,0)			Yes	NULL	
7	kd_pegawai	int(10)			Yes	NULL	

Gambar 28. Struktur Tabel Layanan Beli Obat Khitan

Penjelasan:

Pembuatan tabel di atas bernama `layanan_beli_obat`, terdiri dari 6 atribut data yaitu atribut `kd_beli_obat` bertipe data int(6) dan **auto_increment,** atribut `kd_daftar` bertipe data int(6), atribut `kd_obat` bertipe data int(6), atribut `tgl_layanan bertipe data ` date, atribut `jumlah` bertipe data int(10), `biaya` bertipe data decimal(15,0), dan atribut `kd_pegawai` bertipe data int(10). Data default dalam tabel `layanan_beli_obat`dapat bernilai **NULL** kecuali pada atribut `kd_beli_obat`. Atribut `kd_beli_obat` juga berlaku sebagai kode unik pada tabel `layanan_beli_obat`. Atribut `kd_daftar` berlaku sebagai kode yang bereferensi dari tabel `pendaftaran_pasien`. Atribut `kd_obat` berlaku sebagai kode yang bereferensi dari tabel 'obat`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel 'pegawai'.

7. Membuat Tabel Pengadaan Alat Khitan

Query SQL:

```
CREATE TABLE `pengadaan_alat` (
  `kd_pengadaan_alat` int(6) NOT NULL AUTO_INCREMENT,
  `kd_alat` int(6) DEFAULT NULL,
  `kd_suplier` int(6) DEFAULT NULL,
  `tgl_pengadaan` date DEFAULT NULL,
  `nama_alat` varchar(100) DEFAULT NULL,
  `jumlah` int(10) DEFAULT NULL,
  `harga_beli` decimal(15,0) DEFAULT NULL,
  `harga_jual` decimal(15,0) DEFAULT NULL,
  `total_bayar` decimal(15,0) DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  PRIMARY KEY (`kd_pengadaan_alat`),
  KEY `kd_alat` (`kd_alat`),
  KEY `kd_suplier` (`kd_suplier`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `pengadaan_alat_ibfk_1` FOREIGN KEY (`kd_pegawai`)
  REFERENCES `pegawai` (`kd_pegawai`),
  CONSTRAINT `pengadaan_alat_ibfk_2` FOREIGN KEY (`kd_alat`)
  REFERENCES `alat_khitan` (`kd_alat`),
  CONSTRAINT `pengadaan_alat_ibfk_3` FOREIGN KEY (`kd_suplier`)
  REFERENCES `suplier` (`kd_suplier`)
);
```

Gambar 29. Query Pembuatan Tabel Pengadaan Alat Khitan

Hasil:

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_pengadaan_alat	int(6)			No	None	AUTO_INCREMENT
2	kd_alat	int(6)			Yes	NULL	
3	kd_suplier	int(6)			Yes	NULL	
4	tgl_pengadaan	date			Yes	NULL	
5	nama_alat	varchar(100)	latin1_swedish_ci		Yes	NULL	
6	jumlah	int(10)			Yes	NULL	
7	harga_beli	decimal(15,0)			Yes	NULL	
8	harga_jual	decimal(15,0)			Yes	NULL	
9	total_bayar	decimal(15,0)			Yes	NULL	
10	kd_pegawai	int(10)			Yes	NULL	

Gambar 30. Struktur Tabel Pengadaan Alat Khitan

Penjelasan:

Pembuatan tabel di atas bernama `pengadaan_alat`, terdiri dari 10 atribut data yaitu atribut `kd_pengadaan_alat` bertipe data int(6) dan **auto_increment,** atribut `kd_alat` bertipe data int(6), atribut `kd_suplier` bertipe data int(6)), atribut `tgl_pengadaan` bertipe data date, atribut `nama_alat` bertipe data varchar(100), `jumlah` bertipe data int(10), `harga_beli` bertipe data decimal(15,0), atribut harga_jual bertipe data decimal(15,0), atribut `total_bayar` bertipe data decimal(15,0), dan kd_pegawai` bertipe data int(10). Data default dalam tabel `pengadaan_alat` dapat bernilai **NULL** jika kosong/ tidak diisi, kecuali pada atribut `kd_pengadaan_alat`. Atribut `kd_pengadaan_alat` juga berlaku sebagai kode unik pada tabel `pengadaan_alat`. Atribut `kd_alat` berlaku sebagai kode referensi dari tabel alat_khitan. Atribut `kd_suplier` berlaku sebagai kode referensi dari tabel `suplier`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel ‘pegawai’.

8. Membuat Tabel Pengadaan Obat Khitan

Query SQL:

```
CREATE TABLE `pengadaan_obat` (
  `kd_pengadaan_obat` int(6) NOT NULL AUTO_INCREMENT,
  `kd_obat` int(6) DEFAULT NULL,
  `kd_suplier` int(6) DEFAULT NULL,
  `tgl_pengadaan` date DEFAULT NULL,
  `nama_obat` varchar(100) DEFAULT NULL,
  `jumlah` int(10) DEFAULT NULL,
  `harga_beli` decimal(15,0) DEFAULT NULL,
  `harga_jual` decimal(15,0) DEFAULT NULL,
  `total_bayar` decimal(15,0) DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  PRIMARY KEY (`kd_pengadaan_obat`),
  KEY `kd_obat` (`kd_obat`),
  KEY `kd_suplier` (`kd_suplier`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `pengadaan_obat_ibfk_1` FOREIGN KEY (`kd_obat`)
  REFERENCES `obat` (`kd_obat`),
  CONSTRAINT `pengadaan_obat_ibfk_2` FOREIGN KEY (`kd_suplier`)
  REFERENCES `suplier` (`kd_suplier`),
  CONSTRAINT `pengadaan_obat_ibfk_3` FOREIGN KEY (`kd_pegawai`)
  REFERENCES `pegawai` (`kd_pegawai`)
);
```

Gambar 31. Query Pembuatan Tabel Pengadaan Obat Khitan

Hasil:

Server: 127.0.0.1 » Database: rumah_khitan » Table: pengadaan_obat

Browse | Structure | SQL | Search | Insert | Export | Import | Pr

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_pengadaan_obat	int(6)			No	None	AUTO_INCREMENT
2	kd_obat	int(6)			Yes	NULL	
3	kd_suplier	int(6)			Yes	NULL	
4	tgl_pengadaan	date			Yes	NULL	
5	nama_obat	varchar(100)	latin1_swedish_ci		Yes	NULL	
6	jumlah	int(10)			Yes	NULL	
7	harga_beli	decimal(15,0)			Yes	NULL	
8	harga_jual	decimal(15,0)			Yes	NULL	
9	total_bayar	decimal(15,0)			Yes	NULL	
10	kd_pegawai	int(10)			Yes	NULL	

Gambar 32. Struktur Tabel Pengadaan Obat Khitan

Penjelasan:

Pembuatan tabel di atas bernama `pengadaan_alat`, terdiri dari 10 atribut data yaitu atribut `kd_pengadaan_obat` bertipe data int(6) dan **auto_increment,** atribut `kd_obat` bertipe data int(6), atribut `kd_suplier` bertipe data int(6)), atribut `tgl_pengadaan` bertipe data date, atribut `nama_obat` bertipe data varchar(100),

atribut `jumlah` bertipe data int(10), atribut `harga_beli` bertipe data decimal(15,0), atribut `harga_jual` bertipe data decimal(15,0), atribut dan `total_bayar` bertipe data decimal(15,0), dan atribut 'kd_pegawai` bertipe data int(10). Data default dalam tabel `pengadaan_obat` dapat bernilai **NULL** jika kosong/ tidak diisi, kecuali pada atribut `kd_pengadaan_obat`. Atribut `kd_pengadaan_obat` juga berlaku sebagai kode unik pada tabel `pengadaan_alat`. Atribut `kd_obat` berlaku sebagai kode referensi dari tabel obat. Atribut `kd_suplier` berlaku sebagai kode referensi dari tabel `suplier`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel 'pegawai'.

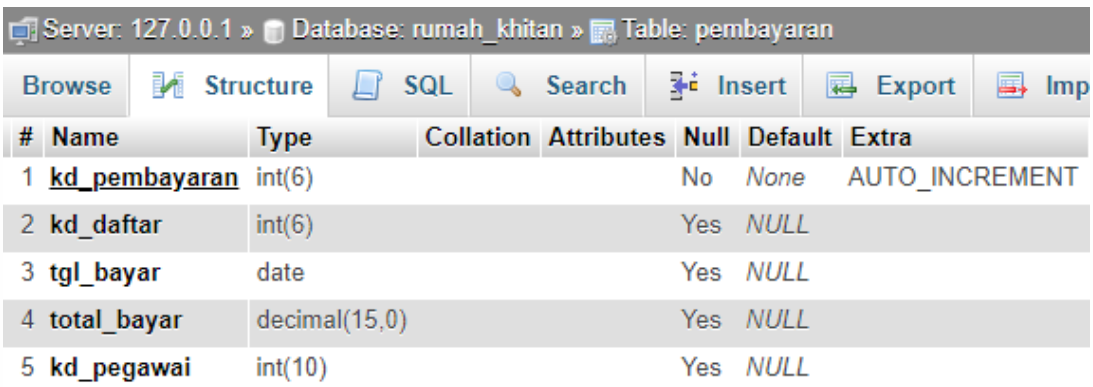
9. Membuat Tabel Pembayaran

   Query SQL:

```
CREATE TABLE `pembayaran` (
  `kd_pembayaran` int(6) NOT NULL AUTO_INCREMENT,
  `kd_daftar` int(6) DEFAULT NULL,
  `tgl_bayar` date DEFAULT NULL,
  `total_bayar` decimal(15,0) DEFAULT NULL,
  `kd_pegawai` int(10) DEFAULT NULL,
  PRIMARY KEY (`kd_pembayaran`),
  KEY `kd_daftar` (`kd_daftar`),
  KEY `kd_pegawai` (`kd_pegawai`),
  CONSTRAINT `pembayaran_ibfk_1` FOREIGN KEY (`kd_pegawai`)
  REFERENCES `pegawai` (`kd_pegawai`),
  CONSTRAINT `pembayaran_ibfk_2` FOREIGN KEY (`kd_daftar`)
  REFERENCES `pendaftaran_pasien` (`kd_daftar`)
);
```

Gambar 33. Query Pembuatan Tabel Pembayaran

Hasil:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_pembayaran	int(6)			No	None	AUTO_INCREMENT
2	kd_daftar	int(6)			Yes	NULL	
3	tgl_bayar	date			Yes	NULL	
4	total_bayar	decimal(15,0)			Yes	NULL	
5	kd_pegawai	int(10)			Yes	NULL	

Gambar 34. Struktur Tabel Pembayaran

Penjelasan:

Pembuatan tabel di atas bernama `pembayaran`, terdiri dari 5 atribut data yaitu atribut `kd_pembayaran` bertipe data int(6) dan **auto_increment,** atribut `kd_daftar` bertipe data int(6), atribut `tgl_bayar` bertipe data date, atribut `total_bayar` bertipe data decimal(15,0) dan atribut `kd_pegawai` bertipe data int(10). Data default dalam tabel `pembayaran` dapat bernilai **NULL** kecuali pada atribut `kd_pembayaran`. Atribut `kd_pembayaran` juga berlaku sebagai kode unik pada tabel `pembayaran`. Atribut `kd_daftar` berlaku sebagai kode bereferensi dari tabel `pembayaran`. Atribut `kd_pegawai` berlaku sebagai kode yang bereferensi dari tabel 'pegawai'.

10. Membuat Tabel Pembayaran Detail

Query SQL:

```
CREATE TABLE `pembayaran_detail` (
  `kd_pembayaran_detail` int(6) NOT NULL AUTO_INCREMENT,
  `kd_pembayaran` int(6) DEFAULT NULL,
  `kd_metode_pembayaran` int(6) DEFAULT NULL,
  `status_bayar` enum('Belum Lunas','Lunas') DEFAULT NULL,
  PRIMARY KEY (`kd_pembayaran_detail`),
  KEY `kd_pembayaran` (`kd_pembayaran`),
  KEY `kd_metode_pembayaran` (`kd_metode_pembayaran`),
  CONSTRAINT `pembayaran_detail_ibfk_1` FOREIGN KEY (`
  kd_pembayaran`) REFERENCES `pembayaran` (`kd_pembayaran`),
  CONSTRAINT `pembayaran_detail_ibfk_2` FOREIGN KEY (`
  kd_metode_pembayaran`) REFERENCES `metode_pembayaran` (`
  kd_metode_pembayaran`)
);
```

Gambar 35. Query Pembuatan Tabel Pembayaran Detail

Hasil:

Server: 127.0.0.1 » Database: rumah_khitan » Table: pembayaran_detail

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Op

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	kd_pembayaran_detail	int(6)			No	None	AUTO_INCREMENT
2	kd_pembayaran	int(6)			Yes	NULL	
3	kd_metode_pembayaran	int(6)			Yes	NULL	
4	status_bayar	enum('Belum Lunas', 'Lunas')	latin1_swedish_ci		Yes	NULL	

Gambar 36. Struktur Tabel Pembayaran Detail

Penjelasan:

Pembuatan tabel di atas bernama `pembayaran_detail`, terdiri dari 4 atribut data yaitu atribut `kd_pembayaran_detail` bertipe data int(6) dan **auto_increment,** atribut `kd_pembayaran` bertipe data int(6), atribut `kd_metode_pembayaran` bertipe data int(6), dan A `status_bayar` bertipe data enum deengan pilihan ('Belum Lunas','Lunas'). Data default dalam tabel `pembayaran_detail` dapat bernilai **NULL** kecuali pada atribut `kd_pembayaran_detail`. Atribut `kd_pembayaran_detail` juga berlaku sebagai kode unik pada tabel. Atribut `kd_pembayaran` berlaku sebagai kode yang bereferensi dari tabel `pembayaran`. Atribut `kd_metode_pembayaran` berlaku sebagai kode yang bereferensi dari tabel `metode_pembayaran`.

C. Pembuatan *Trigger*

Name	Table	Action	Time	Event
t_layanan_beli_alat_ai	layanan_beli_alat	Edit Export Drop	AFTER	INSERT
t_layanan_beli_alat_bi	layanan_beli_alat	Edit Export Drop	BEFORE	INSERT
t_layanan_beli_obat_ai	layanan_beli_obat	Edit Export Drop	AFTER	INSERT
t_layanan_beli_obat_bi	layanan_beli_obat	Edit Export Drop	BEFORE	INSERT
t_layanan_khitan_ai	layanan_khitan	Edit Export Drop	AFTER	INSERT
t_layanan_khitan_bi	layanan_khitan	Edit Export Drop	BEFORE	INSERT
t_layanan_konsultasi_ai	layanan_konsultasi	Edit Export Drop	AFTER	INSERT
t_layanan_konsultasi_bi	layanan_konsultasi	Edit Export Drop	BEFORE	INSERT
t_obat_bi	obat	Edit Export Drop	BEFORE	INSERT
t_pasien_bi	pasien	Edit Export Drop	BEFORE	INSERT
t_pegawai_bi	pegawai	Edit Export Drop	BEFORE	INSERT
t_pembayaran_ai	pembayaran	Edit Export Drop	AFTER	INSERT
t_pembayaran_bu	pembayaran	Edit Export Drop	BEFORE	UPDATE
t_pendaftaran_pasien_ai	pendaftaran_pasien	Edit Export Drop	AFTER	INSERT
t_pendaftaran_pasien_au	pendaftaran_pasien	Edit Export Drop	AFTER	UPDATE
t_pendaftaran_pasien_bi	pendaftaran_pasien	Edit Export Drop	BEFORE	INSERT
t_pengadaan_alat_ai	pengadaan_alat	Edit Export Drop	AFTER	INSERT
t_pengadaan_alat_bi	pengadaan_alat	Edit Export Drop	BEFORE	INSERT
t_pengadaan_obat_ai	pengadaan_obat	Edit Export Drop	AFTER	INSERT
t_pengadaan_obat_bi	pengadaan_obat	Edit Export Drop	BEFORE	INSERT
t_suplier_bi	suplier	Edit Export Drop	BEFORE	INSERT

Gambar 37. Daftar *Trigger Database* Rumah Khitan

Trigger adalah Query SQL yang dijalankan secara otomatis ketika terjadi modifikasi suatu data pada tabel tertentu. Modifikasi data tersebut berupa perintah event INSERT, UPDATE, dan DELETE. *Trigger* memanfaatkan eksekusi SQL berdasarkan kondisi tertentu yaitu BEFORE atau AFTER event.

Trigger pada database rumah khitan ini berjumlah 21. Berikut ini query pembuatan dan penjelasan masing-masing *trigger* yang dibuat:

1. Membuat *Trigger* **t_layanan_beli_alat_ai** pada tabel **layanan_beli_alat**

   Query SQL:

```
/* Trigger structure for table `layanan_beli_alat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_beli_alat_ai`$$
CREATE TRIGGER `t_layanan_beli_alat_ai`
AFTER INSERT ON `layanan_beli_alat`
FOR EACH ROW BEGIN
-- mengurangi jumlah persediaan alat
UPDATE alat_khitan
SET jumlah = jumlah - NEW.jumlah
WHERE kd_alat = NEW.kd_alat;
-- Memperbarui nilai total_bayar (pada tabel pendaftaran_pasien),
UPDATE pendaftaran_pasien
SET total_bayar = total_bayar + NEW.biaya
WHERE kd_daftar = NEW.kd_daftar;
END$$
```

Gambar 38. Query Pembuatan *Trigger* Layanan Beli Alat After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_beli_alat_ai**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_beli_alat_ai** kembali. *Trigger* akan berjalan ototmatis setelah adanya proses insert data pada tabel `layanan_beli_alat` dan berlaku untuk setiap baris data. Isi *trigger* berupa pengurangan jumlah persediaan alat karena telah dibeli pasien dan pembaruan nilai total bayar pada `pendaftaran_pasien`. Pengurangan jumlah persediaan alat dilakukan dengan memperbarui tabel ‘alat_khitan’, kemudian mengatur nilai jumlah sama dengan jumlah sebelumnya dikurangi jumlah sekarang pada pembelian berdasarkan `kd_alat` tertentu yang dibeli. Pembaruan nilai `total_bayar` dilakukan dengan memperbarui tabel ‘pendaftaran_pasien’, kemudian mengatur nilai `total_bayar` sama dengan `total_bayar` sebelumnya ditambah biaya sekarang pada pembelian berdasarkan `kd_daftar` pasien.

2. Membuat *Trigger* **t_layanan_beli_alat_bi** pada tabel **layanan_beli_alat**

Query SQL:

```
/* Trigger structure for table `layanan_beli_alat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_beli_alat_bi`$$
CREATE TRIGGER `t_layanan_beli_alat_bi`
BEFORE INSERT ON `layanan_beli_alat`
FOR EACH ROW BEGIN
-- Mengatur nilai biaya, harus sama dengan jumlah dikali harga
DECLARE v_harga DECIMAL(15);
SELECT F_harga_alat(NEW.kd_alat) INTO v_harga;
SET NEW.biaya = (NEW.jumlah*v_harga);
END$$
```

Gambar 39. Query Pembuatan *Trigger* Layanan Beli Alat Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_beli_alat_bi.** Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_beli_alat_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel `layanan_beli_alat` dan berlaku untuk setiap baris data. Isi *trigger* berupa pengaturan biaya layanan beli alat. Harga alat diperoleh dengan fungsi `F_harga_alat` (penjelasan lihat di bab *function*). Nilai dari fungsi tersebut digunakan untuk mengatur biaya yang baru, dengan cara dikalikan dengan jumlah pembelian alat.

3. Membuat *Trigger* **t_layanan_beli_obat_ai** pada tabel **layanan_beli_obat**

Query SQL:

```
/* Trigger structure for table `layanan_beli_obat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_beli_obat_ai`$$
CREATE TRIGGER `t_layanan_beli_obat_ai`
AFTER INSERT ON `layanan_beli_obat`
FOR EACH ROW BEGIN
-- mengurangi jumlah persediaan obat
UPDATE obat
SET jumlah = jumlah - NEW.jumlah
WHERE kd_obat = NEW.kd_obat;
-- Memperbarui nilai total_bayar (pada tabel pendaftaran_pasien)
UPDATE pendaftaran_pasien
SET total_bayar = total_bayar + NEW.biaya
WHERE kd_daftar = NEW.kd_daftar;
END$$
```

Gambar 40. Query Pembuatan *Trigger* Layanan Beli Obat After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_beli_obat_ai.** Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_beli_obat_ai** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel `layanan_beli_obat` dan berlaku untuk setiap baris data. Isi *trigger* berupa pengurangan jumlah persediaan obat karena telah dibeli pasien dan pembaruan nilai total bayar pada `pendaftaran_pasien`. Pengurangan jumlah persediaan obat dilakukan dengan memperbarui tabel 'obat', kemudian mengatur nilai jumlah sama dengan jumlah sebelumnya dikurangi jumlah sekarang pada pembelian berdasarkan `kd_obat` tertentu yang dibeli. Pembaruan nilai `total_bayar` dilakukan dengan memperbarui tabel 'pendaftaran_pasien', kemudian mengatur nilai `total_bayar` sama dengan `total_bayar` sebelumnya ditambah biaya sekarang pada pembelian berdasarkan `kd_daftar` pasien.

4. Membuat *Trigger* **t_layanan_beli_obat_bi** pada tabel **layanan_beli_obat**

Query SQL:

```
/* Trigger structure for table `layanan_beli_obat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_beli_obat_bi`$$
CREATE TRIGGER `t_layanan_beli_obat_bi`
BEFORE INSERT ON `layanan_beli_obat`
FOR EACH ROW BEGIN
-- Mengatur nilai biaya, harus sama dengan jumlah dikali harga
DECLARE v_harga DECIMAL(15);
SELECT F_harga_obat(NEW.kd_obat) INTO v_harga;
SET NEW.biaya = (NEW.jumlah*v_harga);
END$$
```

Gambar 41. Query Pembuatan *Trigger* Layanan Beli Obat Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_beli_obat_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_beli_obat_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel `layanan_beli_obat` dan berlaku untuk setiap baris data. Isi *trigger* berupa pengaturan biaya layanan beli obat. Harga obat diperoleh dengan fungsi `F_harga_obat` (penjelasan lihat di bab *function*). Nilai

dari fungsi tersebut digunakan untuk mengatur biaya yang baru, dengan cara dikalikan dengan jumlah pembelian obat.

5. Membuat *Trigger* **t_layanan_khitan_ai** pada tabel **layanan_khitan**

   Query SQL:

```
/* Trigger structure for table `layanan_khitan` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_khitan_ai`$$
CREATE TRIGGER `t_layanan_khitan_ai`
AFTER INSERT ON `layanan_khitan`
FOR EACH ROW BEGIN
-- Memperbarui nilai total_bayar (pada tabel pendaftaran_pasien),
UPDATE pendaftaran_pasien
SET total_bayar = total_bayar + NEW.biaya
WHERE kd_daftar = NEW.kd_daftar;
END$$
```

Gambar 42. Query Pembuatan *Trigger* Layanan Khitan After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_khitan_ai**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_khitan_ai** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel `layanan_khitan` dan berlaku untuk setiap baris data. Isi *trigger* berupa pembaruan nilai total bayar pada `pendaftaran_pasien`. Pembaruan nilai `total_bayar` dilakukan dengan memperbarui tabel ‘pendaftaran_pasien’, kemudian mengatur nilai `total_bayar` sama dengan `total_bayar` sebelumnya ditambah biaya sekarang pada layanan khitan berdasarkan `kd_daftar` pasien.

6. Membuat *Trigger* **t_layanan_khitan_bi** pada tabel **layanan_khitan**

Query SQL:

```
/* Trigger structure for table `layanan_khitan` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_khitan_bi`$$
CREATE TRIGGER `t_layanan_khitan_bi`
BEFORE INSERT ON `layanan_khitan`
FOR EACH ROW BEGIN
-- Deklarasi variabel
DECLARE v_biaya DECIMAL(15);
-- Pengecekan kd_daftar, Layanan khitan sebagai layanan utama hanya 1x seumur hidup untuk 1 kd_daftar unik.
-- Jika kd_daftar sudah pernah terdata maka akan muncul "Peringatan: kd_daftar sudah terdata. Layanan khitan 1x se
IF (NEW.kd_daftar IN (SELECT kd_daftar FROM layanan_khitan)) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: kd_daftar sudah terdata. Layanan khitan 1x seumur hidup untuk 1 kd_daftar unik!.";
END IF;
-- Mengatur nilai biaya, harus sama dengan nilai biaya_khitan (dari tabel metode_khitan)
SELECT F_biaya_khitan(NEW.kd_metode_khitan) INTO v_biaya;
SET NEW.biaya = v_biaya;
END$$
```

Gambar 43. Query Pembuatan *Trigger* Layanan Khitan Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_khitan_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_khitan_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel `layanan_khitan` dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan `kd_daftar` dan pengaturan nilai biaya layanan khitan. Pengecekan `kd_daftar` pasien hanya dapat memperoleh layanan khitan 1 kali di rumah khitan. Jika `kd_daftar' terdata di tabel layanan_khitan maka muncul peringatan bahwa `kd_daftar' sudah terdata sebelumnya. Untuk pengaturan nilai biaya layanan khitan harus sama dengan nilai biaya_khitan dari tabel `metode_khitan`. Nilai tersebut diperoleh dengan select fungsi `F_biaya_khitan` (penjelasan lihat di bab *function*). Nilai dari fungsi tersebut digunakan untuk mengatur biaya pada tabel layanan khitan secara ototmatis.

7. Membuat *Trigger* **t_layanan_konsultasi_ai** pada tabel **layanan_konsultasi**

   Query SQL:

```
/* Trigger structure for table `layanan_konsultasi` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_konsultasi_ai`$$
CREATE TRIGGER `t_layanan_konsultasi_ai`
AFTER INSERT ON `layanan_konsultasi`
FOR EACH ROW BEGIN
-- Memperbarui nilai total_bayar (pada tabel pendaftaran_pasien)
UPDATE pendaftaran_pasien
SET total_bayar = total_bayar + NEW.biaya
WHERE kd_daftar = NEW.kd_daftar;
END$$
```

Gambar 44. Query Pembuatan *Trigger* Layanan Konsultasi After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_konsultasi_ai**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_konsultasi_ai** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel `layanan_konsultasi` dan berlaku untuk setiap baris data. Isi *trigger* berupa pembaruan nilai total bayar pada `pendaftaran_pasien`. Pembaruan nilai `total_bayar` dilakukan dengan memperbarui tabel 'pendaftaran_pasien', kemudian mengatur nilai `total_bayar` sama dengan `total_bayar` sebelumnya ditambah biaya sekarang pada layanan konsultasi berdasarkan `kd_daftar` pasien.

8. Membuat *Trigger* **t_layanan_konsultasi_bi** pada tabel **layanan_konsultasi**

   Query SQL:

```
/* Trigger structure for table `layanan_konsultasi` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_layanan_konsultasi_bi`$$
CREATE TRIGGER `t_layanan_konsultasi_bi`
BEFORE INSERT ON `layanan_konsultasi`
FOR EACH ROW BEGIN
-- Pengecekan biaya, harus diisi nilainya.
IF (NEW.biaya = NULL) OR
(NEW.biaya REGEXP '[a-zA-Z ]') OR (NEW.biaya NOT REGEXP '[0-9]') THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Isikan nilai biaya layanan. Hanya dengan bilangan.';
END IF;
END$$
```

Gambar 45. Query Pembuatan *Trigger* Layanan Konsultasi Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_layanan_konsultasi_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_layanan_konsultasi_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel `layanan_konsultasi` dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan biaya layanan harus diisi nilainya berupa angka. Jika biaya layanan NULL atau berupa (karakter huruf a-z, A-Z dan spasi) atau berupa karakter selain angka 0-9 maka akan muncul peringatan untuk mengisikan biaya layanan berupa bilangan.

9. Membuat *Trigger* **t_obat_bi** pada tabel **obat**

Query SQL:

```
/* Trigger structure for table `obat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_obat_bi`$$
CREATE TRIGGER `t_obat_bi`
BEFORE INSERT ON `obat`
FOR EACH ROW BEGIN
-- Pengecekan jenis obat, harus diisi pilihan (Pil/Kapsul/Salep/Lainnya) atau
-- jika tidak tahu jenisnya diisi saja (Belum dipilih).
IF (NEW.jenis NOT IN ('Pil','Kapsul','Salep','Lainnya','Belum dipilih')) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Jenis obat harus diisi pilihan (Pil/Kapsul/Salep/Lainnya),
atau jika tidak tahu jenisnya diisi saja (Belum dipilih)!.";
END IF;
-- Pengecekan kategori obat, harus diisi pilihan (Bebas/Bebas Terbatas/Keras) atau
-- jika tidak tahu kategorinya diisi saja (Belum dipilih).
IF (NEW.kategori NOT IN ('Bebas','Bebas Terbatas','Keras','Belum dipilih')) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Kategori obat harus diisi pilihan (Bebas/Bebas Terbatas/Keras),
atau jika tidak tahu kategorinya diisi saja (Belum dipilih)!.";
END IF;
END$$
```

Gambar 46. Query Pembuatan *Trigger* Obat Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_obat_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_obat_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel obat dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan `jenis_obat` dan pegecekan `kategori` obat. Jenis obat harus diisi denga pilihan (‘Pil’, ’Kapsul’, ’Salep’, ’Lainnya’, ’Belum dipilih’). Jika tidak tahu jenis obat maka isi saja dengan ‘Belum dipilih’. Untuk kategori obat harus diisi (‘Bebas’, ’Bebas

Terbatas', 'Keras', 'Belum dipilih'). Jika tidak tahu kategori obat maka isi saja dengan 'Belum dipilih'.

10. Membuat *Trigger* **t_pasien_bi** pada tabel **pasien**

Query SQL:

```
/* Trigger structure for table `pasien` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pasien_bi`$$
CREATE TRIGGER `t_pasien_bi`
BEFORE INSERT ON `pasien`
FOR EACH ROW BEGIN
-- Pengecekan nomor telepon, tidak boleh kosong harus diisi angka
IF (NEW.telepon REGEXP '[a-zA-Z ]') OR (NEW.telepon NOT REGEXP '[0-9]') THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Isikan nomor telepon dengan angka!.";
END IF;
END$$
```

Gambar 47. Query Pembuatan *Trigger* Pasien Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pasien_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pasien_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel pasien dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan nomor telepon tidak boleh kosong dan harus diisi angka. Jika telepon berupa (karakter huruf a-z, A-Z dan spasi) atau berupa karakter selain angka 0-9 maka akan muncul peringatan untuk mengisikan nomor telepon dengan angka.

11. Membuat *Trigger* **t_pegawai_bi** pada tabel **pegawai**

Query SQL:

```
/* Trigger structure for table `pegawai` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pegawai_bi`$$
CREATE TRIGGER `t_pegawai_bi`
BEFORE INSERT ON `pegawai`
FOR EACH ROW BEGIN
-- Pengecekan jabatan, harus diisi (Kepala Rumah Khitan, Dokter, Perawat, Apoteker, Administrator, atau Bendahara).
IF (NEW.jabatan NOT IN ('Kepala Rumah Khitan','Dokter','Perawat','Apoteker','Administrator','Bendahara')) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Isi jabatan pegawai (Kepala Rumah Khitan,
Dokter, Perawat, Apoteker, Administrator, atau Bendahara)!.";
END IF;
-- Pengecekan jenis_kelamin, harus diisi L untuk laki-laki atau P untuk perempuan
IF (NEW.jenis_kelamin NOT REGEXP '[LP]') THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Jenis kelamin harus diisi L untuk laki-laki atau P untuk perempuan!.";
END IF;
-- Pengecekan nomor telepon, tidak boleh kosong harus diisi angka
IF (NEW.telepon REGEXP '[a-zA-Z ]') OR (NEW.telepon NOT REGEXP '[0-9]') THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Isikan nomor telepon dengan angka!.";
END IF;
END$$
```

Gambar 48. Query Pembuatan *Trigger* Pegawai Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pegawai_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pegawai_bi** kembali. Trigger akan dijalankan ototmatis setelah adanya proses insert data pada tabel pegawai dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan jabatan, pengecekan jenis kelamin, dan pengecekan nomor telepon. Pengecekan jabatan harus diisi (`Kepala Rumah Khitan`, `Dokter`, `Perawat`, `Apoteker`, `Administrator`, atau `Bendahara`). Kemudian, Pengecekan jenis_kelamin, harus diisi L untuk laki-laki atau P untuk perempuan. Pengecekan nomor telepon, tidak boleh kosong harus diisi angka. Jika telepon berupa (karakter huruf a-z, A-Z dan spasi) atau berupa karakter selain angka 0-9 maka akan muncul peringatan untuk mengisikan nomor telepon dengan angka.

12. Membuat *Trigger* **t_pembayaran_ai** pada tabel **pembayaran**

Query SQL:

```
/* Trigger structure for table `pembayaran` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pembayaran_ai`$$
CREATE TRIGGER `t_pembayaran_ai`
AFTER INSERT ON `pembayaran`
FOR EACH ROW BEGIN
-- Tambahkan data kd_pembayaran ke tabel pembayaran_detail
INSERT INTO pembayaran_detail
VALUES ('',NEW.kd_pembayaran,NULL,'Belum Lunas');
END$$
```

Gambar 49. Query Pembuatan *Trigger* Pembayaran After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pembayaran_ai**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pembayaran_ai** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel pembayaran dan berlaku untuk setiap baris data. Isi *trigger* berupa penambahan data kd_pembayaran ke tabel pembayaran detail. Menambahkan data ke tabel pembayaran_detail dengan data (‘’, kd_pembayaran, NULL, ‘Belum Lunas’).

13. Membuat *Trigger* **t_pembayaran_bu** pada tabel **pembayaran**

Query SQL:

```
/* Trigger structure for table `pembayaran` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pembayaran_bu`$$
CREATE TRIGGER `t_pembayaran_bu`
BEFORE UPDATE ON `pembayaran`
FOR EACH ROW BEGIN
-- Pengecekan total_bayar, Jika total_bayar tidak NULL maka
-- muncul "Peringatan: Sudah pernah melakukan pembayaran sebelumnya.
-- Silahkan hanya membayar total_bayar saat ini (tabel pendaftaran_pasien) dikurangi total_bayar sebelumnya (
DECLARE v_total DECIMAL(15);
DECLARE v_total_2 DECIMAL(15);
SELECT total_bayar INTO v_total FROM pembayaran WHERE kd_daftar = OLD.kd_daftar;
SELECT total_bayar INTO v_total_2 FROM pendaftaran_pasien WHERE kd_daftar = OLD.kd_daftar;
IF v_total IS NOT NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Peringatan: Sudah pernah melakukan pembayaran sebelumnya.
        Silahkan update manual total_bayar (tabel pembayaran).';
END IF;
END$$
```

Gambar 50. Query Pembuatan *Trigger* Pembayaran Before Update

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pembayaran_bu**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat trigger

t_pembayaran_bu kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses update data pada tabel pembayaran dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan total bayar. Jika total_bayar tidak NULL maka muncul "Peringatan: Sudah pernah melakukan pembayaran sebelumnya. Silahkan hanya membayar total_bayar saat ini (pada tabel pendaftaran_pasien) dikurangi total_bayar sebelumnya (tabel pembayaran).

14. Membuat *Trigger* **t_pendaftaran_pasien_ai** pada tabel **pendaftaran_pasien**

Query SQL:

```
/* Trigger structure for table `pendaftaran_pasien` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pendaftaran_pasien_ai`$$
CREATE TRIGGER `t_pendaftaran_pasien_ai`
AFTER INSERT ON `pendaftaran_pasien`
FOR EACH ROW BEGIN
-- Tambahkan data ke pendaftaran_pasien_detail
INSERT INTO pendaftaran_pasien_detail
VALUES ('',NEW.kd_daftar,'0000-00-00');
-- Tambahkan data kd_daftar ke tabel pembayaran
INSERT INTO pembayaran
VALUES ('',NEW.kd_daftar,NULL,NULL,NEW.kd_pegawai);
END$$
```

Gambar 51. Query Pembuatan *Trigger* Pendaftaran Pasien After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pendaftaran_pasien_ai**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pendaftaran_pasien_ai** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel pendaftaran_pasien dan berlaku untuk setiap baris data. Isi *trigger* berupa penambahan data baru ke dalam tabel pendaftaran_pasien_detail dan penambahan data `kd_daftar` ke tabel `pembayaran`. Rincian data yang ditambahkan bisa dilihat di Gambar 51.

15. Membuat *Trigger* **t_pendaftaran_pasien_au** pada tabel **pendaftaran_pasien**

Query SQL:

```
/* Trigger structure for table `pendaftaran_pasien` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pendaftaran_pasien_au`$$
CREATE TRIGGER `t_pendaftaran_pasien_au`
AFTER UPDATE ON `pendaftaran_pasien`
FOR EACH ROW BEGIN
-- Pengecekan total_bayar, Jika lebih dari total_bayar sebelumnya maka pembayaran menjadi Belum Lunas
IF NEW.total_bayar > OLD.total_bayar THEN
 UPDATE pembayaran_detail SET status_bayar = 'Belum Lunas'
 WHERE kd_pembayaran = (SELECT kd_pembayaran FROM pembayaran WHERE kd_daftar = OLD.kd_daftar);
END IF;
END$$
```

Gambar 52. Query Pembuatan *Trigger* Pendaftaran Pasien After Update

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pendaftaran_pasien_au**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pendaftaran_pasien_au** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses update data pada tabel pendaftaran_pasien dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan total_bayar baru, jika lebih dari total_bayar sebelumnya maka pembayaran menjadi Belum Lunas. Memperbaru tabel pembayaran_detail dan mengatur status_bayar sama dengan ‘Belum Lunas’ berdasarkan kd_pembayaran (yang ada pada tabel pembayaran berdasarkan kd_daftar pasien tertentu).

16. Membuat *Trigger* **t_pendaftaran_pasien_bi** pada tabel **pendaftaran_pasien**

Query SQL:

```
/* Trigger structure for table `pendaftaran_pasien` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pendaftaran_pasien_bi`$$
CREATE TRIGGER `t_pendaftaran_pasien_bi`
BEFORE INSERT ON `pendaftaran_pasien`
FOR EACH ROW BEGIN
-- Pengecekan kd_pasien, harus harus sudah terdata di tabel pasien.
-- Jika pasien belum terdata maka muncul "Peringatan: Tambahkan data pasien di tabel pasien!.".
IF (NEW.kd_pasien NOT IN (SELECT kd_pasien FROM pasien)) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Tambahkan data pasien di tabel pasien!.";
END IF;
-- Pengecekan kd_pasien, hanya bisa mendaftar 1x. Layanan khitan sebagai layanan utama hanya 1x seumur hidup.
-- Jika pasien sudah pernah terdata maka akan muncul "Peringatan: Pasien sudah pernah mendaftar. Tidak perlu mendaftar kembali!".
IF (NEW.kd_pasien IN (SELECT kd_pasien FROM pendaftaran_pasien)) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Pasien sudah pernah mendaftar.
Tidak perlu mendaftar kembali!.";
END IF;
-- Mengatur nilai total_bayar sebelum insert pendaftaran_pasien, total_bayar di awal pendaftaran = 0 - uang muka.
-- keterangan:
-- nilai + untuk total_bayar yang harus dibayarkan pasien
-- nilai - untuk saldo pengurang total_bayar nantinya
SET NEW.total_bayar = 0-NEW.uang_muka;
END$$
```

Gambar 53. Query Pembuatan *Trigger* Pendaftaran Pasien Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pendaftaran_pasien_bi.** Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pendaftaran_pasien_bi** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel pendaftaran_pasien dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan kd_pasien harus harus sudah terdata di tabel pasien. Jika pasien belum terdata maka muncul "Peringatan: Tambahkan data pasien di tabel pasien!". Kemudian, pengecekan kd_pasien, hanya bisa mendaftar 1x. Layanan khitan sebagai layanan utama hanya 1x seumur hidup. Dan, pengaturan nilai total_bayar sebelum insert pendaftaran_pasien, total_bayar di awal pendaftaran sama dengan 0 dikurangi uang muka. Keterangan besaran total bayar:

a. Nilai + untuk total_bayar yang harus dibayarkan pasien.
b. Nilai - untuk saldo pengurang total_bayar nantinya.

17. Membuat *Trigger* **t_pengadaan_alat_ai** pada tabel **pengadaan_alat**

Query SQL:

```
/* Trigger structure for table `pengadaan_alat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pengadaan_alat_ai`$$
CREATE  TRIGGER `t_pengadaan_alat_ai`
AFTER INSERT ON `pengadaan_alat`
FOR EACH ROW BEGIN
-- Memperbarui data jumlah dan harga alat
-- detail: menambah jumlah persediaan alat dan memperbarui harga
UPDATE alat_khitan SET jumlah=jumlah+New.jumlah WHERE kd_alat = NEW.kd_alat;
UPDATE alat_khitan SET harga = NEW.harga_jual WHERE kd_alat = NEW.kd_alat;
END$$
```

Gambar 54. Query Pembuatan *Trigger* Pengadaan Alat After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pengadaan_alat_ai**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pengadaan_alat_ai** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel pengadaan_alat dan berlaku untuk setiap baris data. Isi *trigger* berupa pembaruan data jumlah dan harga alat. Memperbarui tabel alat_khitan dan mengatur jumlah sama dengan jumlah sebelumnya ditambah jumlah pengadaan alat baru berdasarkan kd_alat tertentu. Dan mengatur harga sama dengan harga jual baru berdasarkan kd_alat tertentu.

18. Membuat *Trigger* **t_pengadaan_alat_bi** pada tabel **pengadaan_alat**

Query SQL:

```
/* Trigger structure for table `pengadaan_alat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pengadaan_alat_bi`$$
CREATE TRIGGER `t_pengadaan_alat_bi`
BEFORE INSERT ON `pengadaan_alat`
FOR EACH ROW BEGIN
-- Pengecekan kd_alat di tabel alat_khitan, jika tidak ada insert data baru
IF NEW.kd_alat NOT IN (SELECT kd_alat FROM alat_khitan) THEN
INSERT INTO alat_khitan VALUES (NEW.kd_alat,NEW.nama_alat,'0','0');
END IF;
-- Mengatur nilai total_bayar, harus sama dengan jumlah dikali harga_beli alat
SET NEW.total_bayar = (NEW.jumlah*NEW.harga_beli);
END$$
```

Gambar 55. Query Pembuatan *Trigger* Pengadaan Alat Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pengadaan_alat_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pengadaan_alat_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel pengadaan_alat dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan kd_alat dan pengaturan nilai total_bayar. Pengecekan kd_alat di tabel alat_khitan, jika tidak ada data sebelumnya maka dilakukan insert data baru (rincian data lihat Gambar 55). Untuk mengatur nilai total_bayar, harus sama dengan jumlah pengadaan alat dikali harga_beli alat.

19. Membuat *Trigger* **t_pengadaan_obat_ai** pada tabel **pengadaan_obat**

Query SQL:

```
/* Trigger structure for table `pengadaan_obat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pengadaan_obat_ai`$$
CREATE TRIGGER `t_pengadaan_obat_ai`
AFTER INSERT ON `pengadaan_obat`
FOR EACH ROW BEGIN
-- Memperbarui data jumlah dan harga obat
-- detail: menambah jumlah persediaan obat dan memperbarui harga
UPDATE obat SET jumlah=jumlah+New.jumlah WHERE kd_obat = NEW.kd_obat;
UPDATE obat SET harga = NEW.harga_jual WHERE kd_obat = NEW.kd_obat;
END$$
```

Gambar 56. Query Pembuatan *Trigger* Pengadaan Obat After Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pengadaan_obat_ai**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pengadaan_obat_ai** kembali. *Trigger* akan dijalankan ototmatis setelah adanya proses insert data pada tabel pengadaan_obat dan berlaku untuk setiap baris data. Isi *trigger* berupa pembaruan data jumlah dan harga obat. Memperbarui tabel obat dan mengatur jumlah sama dengan jumlah sebelumnya ditambah jumlah pengadaan obat baru berdasarkan kd_obat tertentu. Dan mengatur harga sama dengan harga jual baru berdasarkan kd_obat tertentu.

20. Membuat *Trigger* **t_pengadaan_obat_bi** pada tabel **pengadaan_obat**

Query SQL:

```
/* Trigger structure for table `pengadaan_obat` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_pengadaan_obat_bi`$$
CREATE TRIGGER `t_pengadaan_obat_bi`
BEFORE INSERT ON `pengadaan_obat`
FOR EACH ROW BEGIN
-- Pengecekan kode_obat di tabel obat, jika tidak ada insert data baru
IF NEW.kd_obat NOT IN (SELECT kd_obat FROM obat) THEN
INSERT INTO obat VALUES (NEW.kd_obat,NEW.nama_obat,'Belum dipilih','Belum dipilih','0','0')
END IF;
-- Mengatur nilai total_bayar, harus sama dengan jumlah dikali harga_beli obat
SET NEW.total_bayar = NEW.jumlah*NEW.harga_beli;
END$$
```

Gambar 57. Query Pembuatan *Trigger* Pengadaan Obat Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_pengadaan_obat_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_pengadaan_obat_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel pengadaan_obat dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan kd_obat dan pengaturan nilai total_bayar. Pengecekan kd_obat di tabel obat, jika tidak ada data sebelumnya maka dilakukan insert data baru (rincian data lihat Gambar 57). Untuk mengatur nilai total_bayar, harus sama dengan jumlah pengadaan obat dikali harga_beli obat.

21. Membuat *Trigger* **t_suplier_bi** pada tabel **suplier**

Query SQL:

```
/* Trigger structure for table `suplier` */
DELIMITER$$
DROP TRIGGER IF EXISTS `t_suplier_bi`$$
CREATE TRIGGER `t_suplier_bi`
BEFORE INSERT ON `suplier`
FOR EACH ROW BEGIN
-- Pengecekan nomor telepon, tidak boleh kosong harus diisi angka
IF (NEW.telepon REGEXP '[a-zA-Z ]') OR (NEW.telepon NOT REGEXP '[0-9]') THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = "Peringatan: Isikan nomor telepon dengan angka!.";
END IF;
END$$
```

Gambar 58. Query Pembuatan *Trigger* Suplier Before Insert

Penjelasan:

Pembuatan *trigger* di atas bernama **t_suplier_bi**. Query akan menghapus *trigger* jika sudah pernah dibuat. Kemudian, membuat *trigger* **t_suplier_bi** kembali. *Trigger* akan dijalankan ototmatis sebelum adanya proses insert data pada tabel suplier dan berlaku untuk setiap baris data. Isi *trigger* berupa pengecekan nomor telepon, tidak boleh kosong harus diisi angka. Jika telepon berupa (karakter huruf a-z, A-Z dan spasi) atau berupa karakter selain angka 0-9 maka akan muncul peringatan untuk mengisikan nomor telepon dengan angka.

BAB III
STORED PROCEDURE

A. Pengertian *Stored Procedure*

Stored procedure adalah salah sekumpulan query SQL yang sering digunakan dan tersimpan pada database MySQL. Kode SQL dalam *stored procedure* dapat digunakan kembali sewaktu-waktu dengan hanya memanggil nama dari *stored procedure* yang dibuat (Query: **CALL nama_procedure();**) sehingga penggunaan database akan lebih cepat dan efisien.

B. Pembuatan *Stored Procedure*

Stored procedure pada database rumah khitan ini berjumlah 2. Berikut ini query pembuatan dan penjelasan masing-masing *stored procedure* yang dibuat:

1. Membuat Procedure **P_bayar**

   Query SQL:

```
/* Procedure structure for procedure `P_bayar` */
DELIMITER$$
CREATE PROCEDURE `P_bayar`(IN p_kd_daftar INT(6), IN p_tanggal DATE,
IN p_total DECIMAL(15),IN p_metode VARCHAR(50), IN p_kd_pegawai INT(6)) BEGIN
DECLARE kd_bayar INT(6); DECLARE kd_metode INT(6); DECLARE v_status ENUM('Lunas','Belum Lunas'); DECLARE keterangan VARCHAR(200);
SELECT status_bayar INTO v_status FROM pembayaran_detail
WHERE kd_pembayaran = (SELECT kd_pembayaran FROM pembayaran WHERE kd_daftar = p_kd_daftar);
IF v_status = 'Lunas' THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Peringatan: Status bayar sudah Lunas!';
ELSE
        IF (p_kd_daftar IN (SELECT kd_daftar FROM pembayaran)) THEN
                -- Pengecekan total pembayaran, tidak boleh dicicil dan harus sama dengan total bayar di pendaftaran_pasien untuk pelunasan
                IF (p_total = (SELECT total_bayar FROM pendaftaran_pasien WHERE kd_daftar = p_kd_daftar)) THEN
                        -- Pengecekan metode pembayaran, harus ada di metode_pembayaran
                        IF (p_metode IN (SELECT nama_metode FROM metode_pembayaran)) THEN
                                SELECT kd_pembayaran INTO kd_bayar FROM pembayaran WHERE kd_daftar = p_kd_daftar;
                                SELECT kd_metode_pembayaran INTO kd_metode FROM metode_pembayaran WHERE nama_metode = p_metode;
                                -- Memperbarui tgl_bayar pada tabel pembayaran berdasarkan kd_daftar dari pendaftaran_pasien
                                UPDATE pembayaran SET tgl_bayar = p_tanggal WHERE kd_daftar = p_kd_daftar;
                                -- Memperbarui kd_pegawai pada tabel pembayaran berdasarkan kd_daftar dari pendaftaran_pasien
                                UPDATE pembayaran SET kd_pegawai = p_kd_pegawai WHERE kd_daftar = p_kd_daftar;
                                -- Memperbarui kd_metode_pembayaran pada tabel pembayaran_detail berdasarkan kd_pembayaran dari pembayaran
                                UPDATE pembayaran_detail SET kd_metode_pembayaran = kd_metode WHERE kd_pembayaran = kd_bayar;
                                -- Memperbarui status_bayar pada tabel pembayaran_detail berdasarkan kd_pembayaran dari pembayaran
                                UPDATE pembayaran_detail SET status_bayar = 'Lunas' WHERE kd_pembayaran = kd_bayar;
                                SET keterangan = 'Pembayaran berhasil.';
                                -- Memperbarui total_bayar pada tabel pembayaran berdasarkan kd_daftar dari pendaftaran_pasien
                                UPDATE pembayaran SET total_bayar = p_total WHERE kd_daftar = p_kd_daftar;
                        ELSE
                                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Peringatan: Pembayaran Gagal dikarenakan metode pembayaran tidak
                        END IF;
                ELSE SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Peringatan: Pembayaran Gagal dikarenakan pembayaran tidak boleh dicicil dan
                END IF;
        ELSE SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Peringatan: Pembayaran Gagal dikarenakan kd_daftar tidak terdata di tabel pembayara
        END IF;
END IF; SELECT keterangan;
END$$
```

Gambar 59. Query Pembuatan *Procedure* Pembayaran

Penjelasan:

Pembuatan *stored procedure* di atas bernama **P_bayar** dengan parameter IN p_kd_daftar bertipe data INT(6), parameter IN p_tanggal bertipe data DATE, parameter IN p_total bertipe data DECIMAL(15), parameter IN p_metode bertipe data VARCHAR(50), dan parameter IN p_kd_pegawai bertipe data INT(6). Isi *stored procedure* berupa pengecekan status pembayaran. Jika status_bayar sama dengan ‘Lunas’ maka akan muncul 'Peringatan: Status bayar sudah Lunas!'. Jika belum lunas maka dilanjutkan pengecekan kd_daftar dalam tabel pembayaran. Jika tidak ada kd_daftar maka muncul 'Peringatan: Pembayaran Gagal dikarenakan kd_daftar tidak terdata di tabel pembayaran!’. Namun, jika ada kd_daftar dalam tabel pembayaran maka akan dilanjutkan pengecekan total pembayaran. Total pembayaran tidak boleh dicicil dan harus sama dengan total bayar di pendaftaran_pasien untuk pelunasan. Jika tidak sama maka akan muncul 'Peringatan: Pembayaran Gagal dikarenakan pembayaran tidak boleh dicicil dan harus sama dengan total bayar di pendaftaran_pasien untuk pelunasan.'. Jika sama maka akan dilanjutkan pengecekan metode pembayaran, harus ada di metode_pembayaran. Jika tidak ada maka akan muncul 'Peringatan: Pembayaran Gagal dikarenakan metode pembayaran tidak tersedia. Cek tabel metode pembayaran.'. Jika sudah terpenuhi syarat metode pembayaran maka akan dilanjutkan dengan kegiatan berikut:

a. Memperbarui tgl_bayar pada tabel pembayaran berdasarkan kd_daftar dari pendaftaran_pasien.
b. Memperbarui kd_pegawai pada tabel pembayaran berdasarkan kd_daftar dari pendaftaran_pasien.
c. Memperbarui kd_metode_pembayaran pada tabel pembayaran_detail berdasarkan kd_pembayaran dari tabel pembayaran.
d. Memperbarui status_bayar menjadi ‘Lunas’ pada tabel pembayaran_detail berdasarkan kd_pembayaran dari tabel pembayaran.
e. Memperbarui total_bayar pada tabel pembayaran berdasarkan kd_daftar dari pendaftaran_pasien

Jika sudah maka akan muncul keterangan 'Pembayaran berhasil.'.

2. Membuat *Procedure* **P_update_tgl_rencana_khitan**

   Query SQL:

```
/* Procedure structure for procedure `P_update_tgl_rencana_khitan` */
DELIMITER$$
CREATE PROCEDURE `P_update_tgl_rencana_khitan`
(IN p_kode_daftar INT(6),IN p_tanggal DATE)
BEGIN
DECLARE keterangan VARCHAR(200);
IF p_kode_daftar NOT IN (SELECT kd_daftar FROM layanan_khitan) THEN
-- Memperbarui tgl_rencana_khitan pada tabel pendaftaran_pasien_detail berdasarkan kd_daftar
        UPDATE pendaftaran_pasien_detail
        SET tgl_rencana_khitan = p_tanggal
        WHERE kd_daftar = p_kode_daftar;
        SET keterangan = 'Tanggal rencana khitan sudah diperbarui. Silahkan ingatkan pasien
ELSE
        SET keterangan = 'Pasien dengan kode daftar tersebut sudah melaksanakan khitan.';
END IF;
SELECT keterangan;
END$$
```

Gambar 60. Query Pembuatan *Procedure* Update Tanggal Rencana Khitan

Penjelasan:

Stored procedure di atas bernama **P_update_tgl_rencana_khitan** dengan parameter IN p_kode_daftar bertipe data INT(6), dan parameter IN p_tanggal bertipe data DATE. Isi *stored procedure* berupa pembaruan tgl_rencana_khitan pada tabel pendaftaran_pasien_detail berdasarkan kd_daftar dari pendaftaran_pasien. Isi prosedur diawali dengan mengecek kd_daftar pasien, jika terdapat pada tabel layanan khitan maka muncul keterangan bahwa “Pasien dengan kode daftar tersebut sudah melaksanakan khitan”. Jika tidak maka akan dilanjutkan dengan pembaruan tabel pendaftaran_pasien_detail, mengatur tgl_rencana_khitan yang disepakati berdasarkan kd_daftar tertentu. Jika sudah maka akan muncul keterangan bahwa *“Tanggal rencana khitan sudah diperbarui. Silahkan ingatkan pasien untuk datang pada tanggal yang sudah disepakati*”.

BAB IV
FUNCTION

A. Pengertian *Function*

Function merupakan sekumpulan Query SQL pada satu blok yang memiliki konsep seperti *stored procedure*, akan tetapi pada *function* terjadi pengembalian suatu nilai (*return value*). Query yang terkandung sering digunakan sehingga dijadikan suatu *function* untuk meringkas penulisan *script*. *Function* cocok dipakai untuk perhitungan terutama dipakai pada trigger.

B. Pembuatan *Function*

Function pada database rumah khitan berjumlah 15. Berikut ini query pembuatan dan penjelasan masing-masing *function* yang dibuat:

1. Membuat *Function* **F_biaya_khitan**

   Query SQL:

```
/* Function  structure for function  `F_biaya_khitan` */
DELIMITER$$
CREATE FUNCTION `F_biaya_khitan`(p_kd_metode INT(6))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_biaya DECIMAL(15);
SELECT biaya_khitan INTO v_biaya FROM metode_khitan
WHERE kd_metode_khitan = p_kd_metode;
RETURN v_biaya;
END$$
```

Gambar 61. Query Pembuatan *Function* Biaya Khitan

Penjelasan:

Pembuatan *function* di atas bernama **F_biaya_khitan** dengan parameter p_kd_metode bertipe data INT(6) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select biaya_khitan dari tabel metode_khitan berdasarkan `p_kd_metode` terntentu.

2. Membuat *Function* **F_harga_alat**

Query SQL:

```
/* Function  structure for function  `F_harga_alat` */
DELIMITER$$
CREATE FUNCTION `F_harga_alat`(p_kd_alat INT(6))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_harga DECIMAL(15);
SELECT harga INTO v_harga FROM alat_khitan
WHERE kd_alat = p_kd_alat;
RETURN v_harga;
END$$
```

Gambar 62. Query Pembuatan *Function* Harga Alat

Penjelasan:

Pembuatan *function* di atas bernama **F_harga_alat** dengan parameter p_kd_alat bertipe data int(6) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select harga dari tabel alat_khitan berdasarkan `p_kd_alat` terntentu.

3. Membuat *Function* **F_harga_obat**

Query SQL:

```
/* Function  structure for function  `F_harga_obat` */
DELIMITER$$
CREATE FUNCTION `F_harga_obat`(p_kd_obat INT(6))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_harga DECIMAL(15);
SELECT harga INTO v_harga FROM obat
WHERE kd_obat = p_kd_obat;
RETURN v_harga;
END$$
```

Gambar 63. Query Pembuatan *Function* Harga Obat

Penjelasan:

Pembuatan *function* di atas bernama **F_harga_obat** dengan parameter p_kd_obat bertope data INT(6) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select harga dari tabel obat berdasarkan `p_kd_obat` terntentu.

4. Membuat *Function* **F_pemasukan_bulanan**

Query SQL:

```
/* Function  structure for function  `F_pemasukan_bulanan` */
DELIMITER$$
CREATE FUNCTION `F_pemasukan_bulanan`(p_bulan VARCHAR(20))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total FROM pembayaran
WHERE DATE_FORMAT((pembayaran.tgl_bayar),'%m')=p_bulan
OR DATE_FORMAT((pembayaran.tgl_bayar),'%M')=p_bulan;
RETURN v_total;
END$$
```

Gambar 64. Query Pembuatan *Function* Pemasukan Bulanan

Penjelasan:

Pembuatan *function* di atas bernama **F_pemasukan_bulanan** dengan parameter p_bulan bertipe data varchar(20) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pembayaran berdasarkan kondisi p_bulan tertentu.

5. Membuat *Function* **F_pemasukan_total**

Query SQL:

```
/* Function  structure for function  `F_pemasukan_total` */
DELIMITER$$
CREATE FUNCTION `F_pemasukan_total`()
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total FROM pembayaran;
RETURN v_total;
END$$
```

Gambar 65. Query Pembuatan *Function* Pemasukan Total

Penjelasan:

Pembuatan *function* di atas bernama **F_pemasukan_total** tanpa parameter dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pembayaran.

6. Membuat *Function* **F_pengeluaran_alat_bulanan**

Query SQL:

```
/* Function  structure for function  `F_pengeluaran_alat_bulanan` */
DELIMITER$$
CREATE FUNCTION `F_pengeluaran_alat_bulanan`(p_bulan VARCHAR(20))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total_1 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_1 FROM `pengadaan_alat`
WHERE DATE_FORMAT((pengadaan_alat.tgl_pengadaan),'%m')=p_bulan
OR DATE_FORMAT((pengadaan_alat.tgl_pengadaan),'%M')=p_bulan;
RETURN v_total_1;
END$$
```

Gambar 66. Query Pembuatan *Function* Pengeluaran Alat Bulanan

Penjelasan:

Pembuatan *function* di atas bernama **F_pengeluaran_alat_bulanan** dengan parameter p_bulan bertipe data varchar(20) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_alat berdasarkan kondisi p_bulan tertentu.

7. Membuat *Function* **F_pengeluaran_alat_total**

Query SQL:

```
/* Function  structure for function  `F_pengeluaran_alat_total` */
DELIMITER$$
CREATE FUNCTION `F_pengeluaran_alat_total`()
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total_1 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_1 FROM `pengadaan_alat`;
RETURN v_total_1;
END$$
```

Gambar 67. Query Pembuatan *Function* Pengeluaran Alat Total

Penjelasan:

Pembuatan *function* di atas bernama **F_pengeluaran_alat_total** tanpa parameter dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select penjumlahan (dengan fungsi SUM) total_bayar dari tabel `pengadaan_alat`.

8. Membuat *Function* **F_pengeluaran_bulanan**

Query SQL:

```
/* Function  structure for function  `F_pengeluaran_bulanan` */
DELIMITER$$
CREATE FUNCTION `F_pengeluaran_bulanan`(p_bulan VARCHAR(20))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total DECIMAL(15);
DECLARE v_total_1 DECIMAL(15);
DECLARE v_total_2 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_1 FROM `pengadaan_alat`
WHERE DATE_FORMAT((pengadaan_alat.tgl_pengadaan),'%m')=p_bulan
OR DATE_FORMAT((pengadaan_alat.tgl_pengadaan),'%M')=p_bulan;
SELECT SUM(total_bayar) INTO v_total_2 FROM `pengadaan_obat`
WHERE DATE_FORMAT((pengadaan_obat.tgl_pengadaan),'%m')=p_bulan
OR DATE_FORMAT((pengadaan_obat.tgl_pengadaan),'%M')=p_bulan;
SET v_total = v_total_1 + v_total_2;
RETURN v_total;
END$$
```

Gambar 68. Query Pembuatan *Function* Pengeluaran Bulanan

Penjelasan:

Pembuatan *function* di atas bernama **F_pengeluaran_bulanan** dengan parameter p_bulan bertipe data varchar(20) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_alat berdasarkan kondisi p_bulan tertentu) ditambahkan dengan hasil (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_obat berdasarkan kondisi p_bulan tertentu).

9. Membuat *Function* **F_pengeluaran_obat_bulanan**

Query SQL:

```
/* Function  structure for function  `F_pengeluaran_obat_bulanan` */
DELIMITER$$
CREATE FUNCTION `F_pengeluaran_obat_bulanan`(p_bulan VARCHAR(20))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total_2 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_2 FROM `pengadaan_obat`
WHERE DATE_FORMAT((pengadaan_obat.tgl_pengadaan),'%m')=p_bulan
OR DATE_FORMAT((pengadaan_obat.tgl_pengadaan),'%M')=p_bulan;
RETURN v_total_2;
END$$
```

Gambar 69. Query Pembuatan *Function* Pengeluaran Obat Bulanan

Penjelasan:

Pembuatan *function* di atas bernama **F_pengeluaran_obat_bulanan** dengan parameter p_bulan bertipe data varchar(20) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_obat berdasarkan kondisi p_bulan tertentu.

10. Membuat *Function* **F_pengeluaran_obat_total**

Query SQL:

```
/* Function  structure for function  `F_pengeluaran_obat_total` */
DELIMITER$$
CREATE FUNCTION `F_pengeluaran_obat_total`()
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total_2 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_2 FROM `pengadaan_obat`;
RETURN v_total_2;
END$$
```

Gambar 70. Query Pembuatan *Function* Pengeluaran Obat Total

Penjelasan:

Pembuatan *function* di atas bernama **F_pengeluaran_obat_total** tanpa parameter dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_obat.

11. Membuat *Function* **F_pengeluaran_total**

Query SQL:

```
/* Function  structure for function  `F_pengeluaran_total` */
DELIMITER$$
CREATE FUNCTION `F_pengeluaran_total`()
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total_1 DECIMAL(15);
DECLARE v_total_2 DECIMAL(15);
DECLARE v_total_3 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_1 FROM `pengadaan_alat`;
SELECT SUM(total_bayar) INTO v_total_2 FROM `pengadaan_obat`;
SET v_total_3 = v_total_1 + v_total_2;
RETURN v_total_3;
END$$
```

Gambar 71. Query Pembuatan *Function* Pengeluaran Total

Penjelasan:

Pembuatan *function* di atas bernama **F_pengeluaran_total** tanpa parameter dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_alat) ditambahkan dengan hasil (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_obat).

12. Membuat *Function* **F_sisa_bayar**

Query SQL:

```
/* Function  structure for function  `F_sisa_bayar` */
DELIMITER $$
CREATE FUNCTION `F_sisa_bayar`(p_kd_daftar INT(6))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total_now DECIMAL(15);
DECLARE v_total_last DECIMAL(15);
DECLARE v_sisa DECIMAL(15);
SELECT total_bayar INTO v_total_now FROM pendaftaran_pasien WHERE kd_daftar = p_kd_daftar;
SELECT total_bayar INTO v_total_last FROM pembayaran WHERE kd_daftar = p_kd_daftar;
SET v_sisa = v_total_now - v_total_last;
RETURN v_sisa;
END$$
```

Gambar 72. Query Pembuatan *Function* Sisa Pembayaran

Penjelasan:

Pembuatan *function* di atas bernama **F_sisa_bayar** dengan parameter p_kd_daftar bertipe data INT(6) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa pencarian sisa_bayar sama dengan (select total_bayar dari tabel pendaftaran_pasien berdasarkan p_kd_daftar) dikurangi (select total_bayar dari tabel pembayaran berdasarkan p_kd_daftar).

13. Membuat *Function* **F_total_bayar**

Query SQL:

```
/* Function  structure for function  `F_total_bayar` */
DELIMITER$$
CREATE FUNCTION `F_total_bayar`(p_kd_daftar INT(6))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total_now DECIMAL(15);
SELECT total_bayar INTO v_total_now
FROM pendaftaran_pasien
WHERE kd_daftar = p_kd_daftar;
RETURN v_total_now;
END$$
```

Gambar 73. Query Pembuatan *Function* Total Pembayaran

Penjelasan:

Pembuatan *function* di atas bernama **F_total_bayar** dengan parameter p_kd_daftar bertipe data INT(6) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa penampilan nilai total bayar pasien dengan cara select total_bayar dari tabel pendaftaran_pasien berdasarkan p_kd_daftar tertentu.

14. Membuat *Function* **F_untung_rugi_bulanan**

Query SQL:

```
/* Function  structure for function  `F_untung_rugi_bulanan` */
DELIMITER$$
CREATE FUNCTION `F_untung_rugi_bulanan`(p_bulan VARCHAR(20))
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total DECIMAL(15);
DECLARE v_total_1 DECIMAL(15);
DECLARE v_total_2 DECIMAL(15);
DECLARE v_total_3 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_1 FROM pembayaran
WHERE DATE_FORMAT((tgl_bayar),'%m')=p_bulan
OR DATE_FORMAT((tgl_bayar),'%M')=p_bulan;
SELECT SUM(total_bayar) INTO v_total_2 FROM `pengadaan_alat`
WHERE DATE_FORMAT((pengadaan_alat.tgl_pengadaan),'%m')=p_bulan
OR DATE_FORMAT((pengadaan_alat.tgl_pengadaan),'%M')=p_bulan;
SELECT SUM(total_bayar) INTO v_total_3 FROM `pengadaan_obat`
WHERE DATE_FORMAT((pengadaan_obat.tgl_pengadaan),'%m')=p_bulan
OR DATE_FORMAT((pengadaan_obat.tgl_pengadaan),'%M')=p_bulan;
SET v_total = v_total_1 - (v_total_2+v_total_3);
RETURN v_total;
END$$
```

Gambar 74. Query Pembuatan *Function* Untung Rugi Bulanan

Penjelasan:

Pembuatan *function* di atas bernama **F_untung_rugi_bulanan** dengan parameter p_bulan bertipe data varchar(20) dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* diawali dengan mendeklarasikan beberapa variabel pendukung query. Nilai untung rugi bulanan dapat diperoleh dari hasil (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pembayaran berdasarkan kondisi p_bulan tertentu) dikurangi dengan hasil (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_alat berdasarkan kondisi p_bulan tertentu) ditambahkan dengan hasil (select penjumlahan (dengan fungsi

SUM) total_bayar dari tabel pengadaan_obat berdasarkan kondisi p_bulan tertentu).

15. Membuat *Function* **F_untung_rugi_total**

Query SQL:

```
/* Function  structure for function  `F_untung_rugi_total` */
DELIMITER$$
CREATE FUNCTION `F_untung_rugi_total`()
RETURNS DECIMAL(15,0)
BEGIN
DECLARE v_total DECIMAL(15);
DECLARE v_total_1 DECIMAL(15);
DECLARE v_total_2 DECIMAL(15);
DECLARE v_total_3 DECIMAL(15);
SELECT SUM(total_bayar) INTO v_total_1 FROM pembayaran;
SELECT SUM(total_bayar) INTO v_total_2 FROM `pengadaan_alat`;
SELECT SUM(total_bayar) INTO v_total_3 FROM `pengadaan_obat`;
SET v_total = v_total_1 - (v_total_2+v_total_3);
RETURN v_total;
END$$
```

Gambar 75. Query Pembuatan *Function* Untung Rugi Total

Penjelasan:

Pembuatan *function* di atas bernama **F_untung_rugi_total** tanpa parameter dan akan mengembalikan nilai bertipe data decimal(15,0). Isi *function* berupa penghitungan nilai untung rugi total yang diperoleh dari hasil (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pembayaran) dikurangi dengan hasil (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_alat) ditambahkan dengan hasil (select penjumlahan (dengan fungsi SUM) total_bayar dari tabel pengadaan_obat).

BAB V
PENUTUP

A. Kesimpulan

Dari materi panduan praktis membuat database ini kita telah dapat mempraktikkan kegiatan mendesain database yang baik khusunya database rumah khitan. Dalam desain database telah terdapat relasi-relasi antar tabel master dan transaksi yang dengan jelas menggambarkan keterkaitan data antar tabel. Tabel data master berjumlah sebanyak 7 tabel dan tabel transaksi ada 10. Terdapat 21 macam *trigger* yang dibuat untuk pengelolaan sebagian data secara otomatis, serta terdapat 2 *stored procedure* dan 15 *function* untuk perhitungan-perhitungan dalam database. Komputasi fungsi dalam database meliputi:

a) Perhitungan biaya masing-masing layanan (khitan, konsultasi, beli alat, dan beli obat).
b) Perhitungan total pembayaran oleh pasien sama dengan hasil penjumlahan keseluruhan biaya layanan (khitan, konsultasi, beli alat, dan beli obat) pada point a.
c) Terdapat perhitungan persediaan (alat khitan dan obat khitan) yaitu total biaya pengadaan di masing-masing tabel pengadaan, penambahan stok (alat atau obat) setelah pengadaan barang oleh pegawai dan pengurangan stok (alat atau obat) setelah dibeli oleh pasien.
d) Terdapat perhitungan biaya pengeluaran yang diperoleh dari penjumlahan total bayar pengadaan (alat atau obat) dan perhitungan pemasukan yang diperoleh dari penjumlahan totol bayar layanan oleh pasien.
e) Perhitungan untung rugi dalam sistem database rumah khitam diperoleh dari selisih atau hasil pengurangan antara pemasukan dengan biaya pengeluaran.

B. Saran

Desain sistem database rumah khitan ini dapat dikembangkan lagi dengan penambahan tabel-tabel data lain dan dengan relasi tabel yang lebih baik. Tabel data yang dapat ditambahkan seperti penjadwalan karyawan, penggajian pegawai, dll.

REFERENSI

Harrison, Guy. & Feuerstein, Steven. 2006. *MySQL Stored Procedure Programming*. Cambridge: O'Reilly Media, Inc.

LAMPIRAN-LAMPIRAN

LAMPIRAN 01: Tampilan Input Data Keseluruhan pada Database

A. Tampilan Data Tabel Pasien

kd_pasien	nama_pasien	tgl_lahir	alamat	kota	wali_pasien	telepon
1	Riko	2009-08-01	Jalan Melati	Surabaya	Sigit	0897656786
2	Ardian	2009-03-23	Jalan Pemuda	Bangkalan	Subagja	0786666
3	Faruq	2009-03-21	Jalan Melati	Surabaya	Susilo	0896666567
4	Hafiz	2009-08-25	Jalan Jepara	Surabaya	Joko	0876567567
5	Andi	2010-11-12	Jalan Blega	Bangkalan	Bambang	0845664356
6	Susilo	2009-03-11	Jalan Telang	Bangkalan	Ali	086745646
7	Aji	2009-08-21	Jalan Pemuda	Bangkalan	Abdurrohman	0864674674
8	Wahyu	2010-11-09	Jalan Melati	Surabaya	Suyono	0863563
9	Akbar	2009-02-23	Jalan Pemuda	Gresik	Arif	0899823423
10	Ratno	2010-10-21	Jalan Anggrek	Surabaya	Evan	087657886
11	Sukmo	2010-10-07	Jalan Pantura	Gresik	Kamal	086178168
12	Bayu	2009-08-09	Jalan Telang	Bangkalan	Hamzah	087286891
13	Agiel	2009-03-21	Jalan Telang	Bangkalan	Setiawan	0981972816
14	Wijaya	2010-11-21	Jalan Pemuda	Gresik	Fandi	08126812
15	Adi	2010-09-25	Jalan Pemuda	Gresik	Fata	01828127377
16	Ahmad	2010-12-20	Jalan Telang	Bangkalan	Imam	08277727129
17	Yusuf	2009-03-22	Jalan Pantura	Gresik	Ian	0218667631
18	Jaya	2008-03-21	Jalan Telang	Bangkalan	Fiyan	01291877
19	Dika	2010-12-21	Jalan Anggrek	Surabaya	Santoso	08192818
20	Diki	2010-12-21	Jalan Anggrek	Surabaya	Santoso	08192818
21	Bagas	2011-05-05	Jalan Pemuda	Bangkalan	Johan	0824983
22	Bondan	2010-02-07	Jalan Pemuda	Bangkalan	Budi	089897343
23	Angga	2011-01-05	Kenjeran	Surabaya	Ardian	0233242
24	Ilham	2010-02-23	Jalan Telang	Bangkalan	Mahdi	08232323
25	Yusron	2011-09-24	Jalan Semarang	Surabaya	Mahmud	08727932
26	Dhani	2011-02-18	Jalan Telang	Bangkalan	Santo	0876979797
27	Yusril	2010-02-23	Jalan Semarang	Surabaya	Darmo	08289238884

Database: rumah_khitan Table: pasien

Gambar 76. Isi Data Tabel Pasien

B. Tampilan Data Tabel Pegawai

kd_pegawai	nama_pegawai	jabatan	jenis_kelamin	alamat	telepon
1	Hendra Subagja	Kepala Rumah Kh...	L	Surabaya	082111111
2	Dedi Subagja	Dokter	L	Surabaya	082311111
3	Abdur Rouf	Perawat	L	Jepara	083411111
4	Anisa Safitri	Perawat	P	Gresik	08969999
5	Rahmawati	Perawat	P	Gresik	0896777
6	Rendi Satya	Administrator	L	Bangkalan	08654444
7	Sintia Mada	Administrator	P	Bangkalan	0875666
8	Nawawi	Apoteker	L	Surabaya	0986666
9	Vika Sari	Bendahara	P	Bangkalan	08977777
10	Andini	Bendahara	P	Surabaya	0974566666
11	Rifan	Administrator	L	Jepara	087898788898
12	Kiki	Apoteker	P	Gresik	087988788000
13	Rio	Apoteker	L	Surabaya	087988788111
14	Rani	Bendahara	P	Surabaya	087888788222
15	Firman	Bendahara	L	Bangkalan	087988788333
16	Lutfi Aji	Bendahara	L	Bangkalan	085766877900
17	Lutfiana	Bendahara	P	Bangkalan	087890098998
* (Auto)	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*

Gambar 77. Isi Data Tabel Pegawai

C. Tampilan Data Tabel Metode Khitan

kd_metode_khitan	nama_metode	biaya_khitan	keterangan
1	Konvensional	600000	Metode yang sudah ada sejak jaman dulu yai
2	Elektrik Couter (Laser)	600000	Pemotongan atau pengangkatan kulit dengan
3	Smart Clamp	850000	Dengan menggunakan alat tabung yang ditemp
4	Alis Clamp	850000	Dengan menempelkan klem dan posisinya agal
5	Tindakan Khusus	1950000	Metode khitan khusus anak gemuk dengan mer
6	Khitan Dewasa	1700000	Orang dewasa biasanya membutuhkan jahitan
7	Laser CO2	1500000	Laser yang digunakan adalah laser CO2. Set
8	Khitan Stapler	1500000	Stapler adalah metode sunat yang menggunal
9	Shang Ring (Metode Klem)	500000	Shang ring terdiri dari dua cincin plastil
10	Plastibell (Metode Klem)	500000	Plastibell diperuntukkan bayi sampai denga
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*

Gambar 78. Isi Data Tabel Metode Khitan

D. Tampilan Data Tabel Alat Khitan

kd_alat	nama_alat	jumlah	harga
1	Sarung	130	50000
2	Perban	130	5000
3	Kassa Steril	240	5000
4	Sarung Tangan	300	10000
5	Celana Khitan	300	15000
6	Plester	300	5000
7	Kapas	300	10000
8	Tisu	225	10000
9	Gunting	150	15000
10	Pinset	270	15000
11	Tisu Basah	200	15000
12	Alat Suntik	200	15000
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*

Gambar 79. Isi Data Tabel Alat Khitan

E. Tampilan Data Tabel Obat Khitan

kd_obat	nama_obat	jenis	kategori	jumlah	harga
1	Nebacetin	Salep	Keras	213	30000
2	Nebacetin Powder	Lainnya	Keras	209	35000
3	QnC Gamat	Salep	Bebas	300	165000
4	Pureganta	Lainnya	Bebas	300	135000
5	Albucare	Kapsul	Bebas	300	180000
6	ChannaMax Bharata	Kapsul	Bebas	225	275000
7	Sulfanilamide Powder	Lainnya	Bebas Terb...	225	5000
8	Obat Tetes KidTan	Lainnya	Bebas	150	30000
9	Enbatic	Salep	Keras	225	15000
10	Enbatic Powder	Lainnya	Keras	300	5000
11	Betadine	Belum dip...	Belum dipilih	300	15000
12	Alkohol 70%	Belum dip...	Belum dipilih	400	15000
13	Betadine 60ml	Belum dip...	Belum dipilih	250	15000
14	Obat Tetes KidTan 60ml	Belum dip...	Belum dipilih	300	20000
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*

Gambar 80. Isi Data Tabel Obat Khitan

F. Tampilan Data Tabel Suplier

kd_suplier	nama_suplier	alamat	telepon
1	CV Cininta Farma	Surabaya	087888
2	PT Sembada Daya Utama	Surabaya	0878988
3	PT Dwi Sakti Jaya	Jakarta	0876823899
4	CV Mandiri Jaya	Semarang	0897777
5	PT Medika Grup Indo	Surabaya	08655555
6	CV Interindo Sehat	Malang	0875555
7	PT Metro Jaya Sakti	Bandung	0856666
8	CV Bangkit Lestari	Semarang	08965555
9	CV Catur Buana	Surabaya	08988766765
10	CV Citra Lestari Jaya	Semarang	088767888
11	CV Sehat Selalu	Semarang	087600670900
12	CV Farma Indica	Malang	087600670901
13	CV Zola Herbal	Semarang	087600670902
14	PT Dexa Dot Com	Yogyakarta	087600670903
15	CV Eka Pilar	Solo	087600670904
16	PT Dwi Wangsa Group	Solo	087600670905
17	CV Bagus Husada	Bandung	087600670906
18	CV Karsa Medika	Bandung	087600670907
19	CV Mandiri Herbal	Malang	087600670908
20	PT Sembada Farma	Gresik	087600670909
21	PT Afa Health Center	Jepara	08976555788
22	CV Ndeso Care	Jepara	08976555009
23	CV Rija Medika	Jepara	08976555010
24	CV Andika Herbal	Semarang	08976555011
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*

Gambar 81. Isi Data Tabel Suplier

G. Tampilan Data Tabel Metode Pembayaran

kd_metode_pembayaran	nama_metode
1	Tunai
2	Kartu Kredit
3	Kartu Debit
4	Transfer Bank
5	Voucher / Gift Card
6	Jenius
7	Gopay
8	Dana
9	OVO
10	Link Aja
(Auto)	*(NULL)*

Gambar 82. Isi Data Tabel Metode Pembayaran

H. Tampilan Data Tabel Pendaftaran Pasien

kd_daftar	kd_pasien	tgl_daftar	uang_muka	kd_pegawai	total_bayar
1	1	2020-01-02	0	6	960000
2	2	2020-01-02	0	7	1020000
3	3	2020-01-08	0	6	665000
4	4	2020-01-09	0	7	635000
5	5	2020-01-12	0	7	960000
6	6	2020-02-12	0	6	900000
7	7	2020-02-12	0	6	735000
8	8	2020-02-15	0	7	990000
9	9	2020-02-18	0	6	730000
10	10	2020-02-20	0	7	665000
11	11	2020-03-10	0	6	960000
12	12	2020-03-10	0	7	915000
13	13	2020-03-12	0	6	905000
14	14	2020-03-18	0	7	1030000
15	15	2020-03-23	0	6	935000
16	16	2020-04-10	0	7	960000
17	17	2020-04-19	0	6	940000
18	18	2020-04-19	0	6	665000
19	19	2020-04-23	0	7	645000
20	20	2020-04-25	0	7	930000
21	21	2020-05-05	0	7	905000
22	22	2020-05-05	0	6	690000
23	23	2020-05-06	0	6	935000
24	24	2020-05-08	0	7	955000
25	25	2020-05-23	0	6	930000
26	26	2020-06-01	0	6	890000
27	27	2020-06-01	0	7	980000

Database: rumah_khitan Table: pendaftaran_pasien

Gambar 83. Isi Data Tabel Pendaftaran Pasien

I. Tampilan Data Tabel Pendaftaran Pasien Detail

kd_daftar_detail	kd_daftar	tgl_rencana_khitan
1	1	2020-01-02
2	3	2020-01-02
3	3	2020-01-08
4	4	2020-01-09
5	5	2020-01-12
6	6	2020-02-12
7	7	2020-02-12
8	8	2020-02-15
9	9	2020-02-18
10	10	2020-02-20
11	11	2020-03-10
12	12	2020-03-10
13	13	2020-03-12
14	14	2020-03-18
15	15	2020-03-23
16	16	2020-04-10
17	17	2020-04-19
18	18	2020-04-19
19	19	2020-04-23
20	20	2020-04-25
21	21	2020-05-05
22	22	2020-05-05
23	23	2020-05-06
24	24	2020-05-08
25	25	2020-05-23
26	26	2020-06-01
27	27	2020-06-01

Database: rumah_khitan Table: pendaftaran_pasien_detail

Gambar 84. Isi Data Tabel Pendaftaran Pasien Detail

J. Tampilan Data Tabel Layanan Khitan

kd_khitan	kd_daftar	kd_metode_khitan	tgl_khitan	kd_pegawai	biaya
1	1	3	2020-01-02	2	850000
2	2	3	2020-01-02	2	850000
3	3	1	2020-01-08	2	600000
4	4	1	2020-01-09	2	600000
5	5	4	2020-01-12	2	850000
6	6	4	2020-02-12	2	850000
7	7	2	2020-02-12	2	600000
8	8	3	2020-02-15	2	850000
9	9	2	2020-02-18	2	600000
10	10	2	2020-02-20	2	600000
11	11	3	2020-03-10	2	850000
12	12	4	2020-03-10	2	850000
13	13	3	2020-03-12	2	850000
14	14	4	2020-03-18	2	850000
15	15	4	2020-03-23	2	850000
16	16	3	2020-04-10	2	850000
17	17	3	2020-04-19	2	850000
18	18	1	2020-04-19	2	600000
19	19	1	2020-04-23	2	600000
20	20	3	2020-04-25	2	850000
21	21	4	2020-05-05	2	850000
22	22	1	2020-05-05	2	600000
23	23	3	2020-05-06	2	850000
24	24	3	2020-05-08	2	850000
25	25	4	2020-05-23	2	850000
26	26	3	2020-06-01	2	850000
27	27	3	2020-06-01	2	850000

Database: rumah_khitan Table: layanan_khitan

Gambar 85. Isi Data Tabel Layanan Khitan

K. Tampilan Data Tabel Layanan Konsultasi

kd_konsultasi	kd_daftar	tgl_konsultasi	kd_pegawai	biaya
1	1	2020-01-02	3	0
2	2	2020-01-02	3	0
3	3	2020-01-08	3	20000
4	4	2020-01-09	3	0
5	5	2020-01-12	3	0
6	6	2020-02-12	3	10000
7	7	2020-02-12	3	0
8	8	2020-02-15	2	20000
9	9	2020-02-18	2	0
10	10	2020-02-20	3	0
11	11	2020-03-10	3	0
12	12	2020-03-10	3	0
13	13	2020-03-12	3	20000
14	14	2020-03-18	3	10000
15	15	2020-03-23	3	20000
16	16	2020-04-10	2	0
17	17	2020-04-19	4	10000
18	18	2020-04-19	4	20000
19	19	2020-04-23	2	0
20	20	2020-04-25	3	0
21	21	2020-05-05	4	20000
22	22	2020-05-05	3	10000
23	23	2020-05-06	3	10000
24	24	2020-05-08	2	20000
25	25	2020-05-23	3	0
26	26	2020-06-01	3	0
27	27	2020-06-01	2	20000

Database: rumah_khitan **Table:** layanan_konsultasi

Gambar 86. Isi Data Tabel Layanan Konsultasi

L. Tampilan Data Tabel Layanan Beli Alat Khitan

kd_beli_alat	kd_daftar	kd_alat	tgl_layanan	jumlah	biaya	kd_pegawai
1	1	1	2020-01-02	1	50000	10
2	2	1	2020-01-02	2	100000	10
3	3	2	2020-01-08	2	10000	10
4	4	2	2020-01-09	1	5000	10
5	5	1	2020-01-12	1	50000	10
6	6	2	2020-02-12	2	10000	10
7	7	1	2020-02-12	2	100000	10
8	8	1	2020-02-15	1	50000	10
9	9	1	2020-02-18	2	100000	10
10	10	2	2020-02-20	1	5000	10
11	11	1	2020-03-10	1	50000	10
12	12	2	2020-03-10	1	5000	10
13	13	2	2020-03-12	1	5000	10
14	14	1	2020-03-18	2	100000	10
15	15	2	2020-03-23	1	5000	10
16	16	1	2020-04-10	1	50000	10
17	17	1	2020-04-19	1	50000	10
18	18	2	2020-04-19	2	10000	10
19	19	2	2020-04-23	2	10000	10
20	20	1	2020-04-25	1	50000	10
21	21	2	2020-05-05	1	5000	10
22	22	1	2020-05-05	1	50000	10
23	23	2	2020-05-06	1	5000	10
24	24	1	2020-05-08	1	50000	10
25	25	1	2020-05-23	1	50000	10
26	26	2	2020-06-01	1	5000	10
27	27	1	2020-06-01	1	50000	10

Database: rumah_khitan Table: layanan_beli_alat

Gambar 87. Isi Data Tabel Layanan Beli Alat Khitan

M. Tampilan Data Tabel Layanan Beli Obat Khitan

kd_beli_obat	kd_daftar	kd_obat	tgl_layanan	jumlah	biaya	kd_pegawai
1	1	1	2020-01-02	2	60000	10
2	2	2	2020-01-02	2	70000	10
3	3	2	2020-01-08	1	35000	7
4	4	1	2020-01-09	1	30000	7
5	5	1	2020-01-12	2	60000	8
6	6	1	2020-02-12	1	30000	7
7	7	2	2020-02-12	1	35000	7
8	8	2	2020-02-15	2	70000	8
9	9	1	2020-02-18	1	30000	7
10	10	1	2020-02-20	2	60000	8
11	11	1	2020-03-10	2	60000	8
12	12	1	2020-03-10	2	60000	7
13	13	1	2020-03-12	1	30000	8
14	14	2	2020-03-18	2	70000	7
15	15	1	2020-03-23	2	60000	8
16	16	1	2020-04-10	2	60000	8
17	17	1	2020-04-19	1	30000	7
18	18	2	2020-04-19	1	35000	7
19	19	2	2020-04-23	1	35000	8
20	20	1	2020-04-25	1	30000	8
21	21	1	2020-05-05	1	30000	8
22	22	1	2020-05-05	1	30000	8
23	23	2	2020-05-06	2	70000	7
24	24	2	2020-05-08	1	35000	7
25	25	1	2020-05-23	1	30000	7
26	26	2	2020-06-01	1	35000	8
27	27	1	2020-06-01	2	60000	8

Database: rumah_khitan **Table:** layanan_beli_obat

Gambar 88. Isi Data Tabel Layanan Beli Obat Khitan

N. Tampilan Data Tabel Pengadaan Alat Khitan

kd_pengadaan_alat	kd_alat	kd_suplier	tgl_pengadaan	nama_alat	jumlah	harga_beli	harga_jual	total_bayar	kd_pegawai
1	1	2	2020-01-01	Sarung	50	45000	50000	2250000	9
2	2	1	2020-01-01	Perban	50	4000	5000	200000	9
3	3	1	2020-01-02	Kassa Steril	80	4000	5000	320000	9
4	4	1	2020-01-02	Sarung Tangan	100	8000	10000	800000	9
5	5	5	2020-01-02	Celana Khitan	100	10000	15000	1000000	9
6	6	12	2020-02-03	Plester	100	3000	5000	300000	15
7	7	13	2020-02-03	Kapas	100	80000	10000	8000000	15
8	8	14	2020-02-05	Tisu	75	8000	10000	600000	15
9	9	15	2020-02-06	Gunting	50	1000	15000	50000	15
10	10	16	2020-02-07	Pinset	90	13000	15000	1170000	15
11	1	2	2020-03-11	Sarung	50	45000	50000	2250000	9
12	2	1	2020-03-11	Perban	50	4000	5000	200000	9
13	3	1	2020-03-12	Kassa Steril	80	4000	5000	320000	9
14	4	1	2020-03-16	Sarung Tangan	100	8000	10000	800000	9
15	5	5	2020-03-20	Celana Khitan	100	10000	15000	1000000	9
16	6	12	2020-04-03	Plester	100	3000	5000	300000	15
17	7	13	2020-04-05	Kapas	100	80000	10000	8000000	15
18	8	14	2020-04-08	Tisu	75	8000	10000	600000	15
19	9	15	2020-04-09	Gunting	50	1000	15000	50000	15
20	10	16	2020-04-15	Pinset	90	13000	15000	1170000	15
21	1	2	2020-05-11	Sarung	50	45000	50000	2250000	9
22	2	1	2020-05-11	Perban	50	4000	5000	200000	9
23	3	1	2020-05-12	Kassa Steril	80	4000	5000	320000	9
24	4	1	2020-05-16	Sarung Tangan	100	8000	10000	800000	9
25	5	5	2020-05-20	Celana Khitan	100	10000	15000	1000000	9
26	6	12	2020-06-03	Plester	100	3000	5000	300000	15
27	7	13	2020-06-05	Kapas	100	80000	10000	8000000	15

Database: rumah_khitan Table: pengadaan_alat

Gambar 89. Isi Data Tabel Pengadaan Alat Khitan

O. Tampilan Data Tabel Pengadaan Obat Khitan

kd_pengadaan_obat	kd_obat	kd_suplier	tgl_pengadaan	nama_obat	jumlah	harga_beli	harga_jual	total_bayar
1	1	3	2020-01-01	Nebacetin	80	25000	30000	2000000
2	2	3	2020-01-01	Nebacetin Powder	75	30000	35000	2250000
3	3	4	2020-01-02	QnC Gamat	100	150000	165000	15000000
4	4	6	2020-01-02	Pureganta	100	120000	135000	12000000
5	5	7	2020-01-02	Albucare	100	160000	180000	16000000
6	6	8	2020-02-02	ChannaMax Bharata	75	250000	275000	18750000
7	7	9	2020-02-02	Sulfanilamide Powder	75	4000	5000	300000
8	8	10	2020-02-02	Obat Tetes KidTan	50	25000	30000	1250000
9	9	11	2020-02-05	Enbatic	75	10000	15000	750000
10	10	11	2020-02-05	Enbatic Powder	100	4000	5000	400000
11	1	3	2020-03-04	Nebacetin	80	25000	30000	2000000
12	2	3	2020-03-04	Nebacetin Powder	75	30000	35000	2250000
13	3	4	2020-03-08	QnC Gamat	100	150000	165000	15000000
14	4	6	2020-03-10	Pureganta	100	120000	135000	12000000
15	5	7	2020-03-15	Albucare	100	160000	180000	16000000
16	6	8	2020-04-01	ChannaMax Bharata	75	250000	275000	18750000
17	7	9	2020-04-01	Sulfanilamide Powder	75	4000	5000	300000
18	8	10	2020-04-02	Obat Tetes KidTan	50	25000	30000	1250000
19	9	11	2020-04-05	Enbatic	75	10000	15000	750000
20	10	11	2020-04-05	Enbatic Powder	100	4000	5000	400000
21	1	3	2020-05-04	Nebacetin	80	25000	30000	2000000
22	2	3	2020-05-04	Nebacetin Powder	75	30000	35000	2250000
23	3	4	2020-05-08	QnC Gamat	100	150000	165000	15000000
24	4	6	2020-05-10	Pureganta	100	120000	135000	12000000
25	5	7	2020-05-15	Albucare	100	160000	180000	16000000
26	6	8	2020-06-01	ChannaMax Bharata	75	250000	275000	18750000

Database: rumah_khitan Table: pengadaan_obat

Gambar 90. Isi Data Tabel Pengadaan Obat Khitan

P. Tampilan Data Tabel Pembayaran

kd_pembayaran	kd_daftar	tgl_bayar	total_bayar	kd_pegawai
1	1	2020-01-02	960000	15
2	2	2020-01-02	1020000	14
3	3	2020-01-08	665000	15
4	4	2020-01-09	635000	15
5	5	2020-01-12	960000	14
6	6	2020-02-12	900000	14
7	7	2020-02-12	735000	14
8	8	2020-02-15	990000	15
9	9	2020-02-18	730000	15
10	10	2020-02-20	665000	14
11	11	2020-03-10	960000	14
12	12	2020-03-10	915000	14
13	13	2020-03-12	905000	15
14	14	2020-03-18	1030000	15
15	15	2020-03-23	935000	15
16	16	2020-04-10	960000	15
17	17	2020-04-19	940000	15
18	18	2020-04-19	665000	14
19	19	2020-04-23	645000	14
20	20	2020-04-25	930000	14
21	21	2020-05-05	905000	14
22	22	2020-05-05	690000	14
23	23	2020-05-06	935000	14
24	24	2020-05-08	955000	15
25	25	2020-05-23	930000	14
26	26	2020-06-01	890000	15
27	27	2020-06-01	980000	14

Database: rumah_khitan Table: pembayaran

Gambar 91. Isi Data Tabel Pembayaran

Q. Tampilan Data Tabel Pembayaran Detail

kd_pembayaran_detail	kd_pembayaran	kd_metode_pembayaran	status_bayar
1	1	1	Lunas
2	2	3	Lunas
3	3	3	Lunas
4	4	4	Lunas
5	5	1	Lunas
6	6	1	Lunas
7	7	1	Lunas
8	8	1	Lunas
9	9	2	Lunas
10	10	1	Lunas
11	11	1	Lunas
12	12	1	Lunas
13	13	1	Lunas
14	14	1	Lunas
15	15	1	Lunas
16	16	1	Lunas
17	17	1	Lunas
18	18	3	Lunas
19	19	3	Lunas
20	20	3	Lunas
21	21	3	Lunas
22	22	3	Lunas
23	23	4	Lunas
24	24	4	Lunas
25	25	3	Lunas
26	26	1	Lunas
27	27	1	Lunas

Database: rumah_khitan Table: pembayaran_detail

Gambar 92. Isi Data Tabel Pembayaran Detail

LAMPIRAN 02: Identitas Penulis

1. Data Diri
   a. Nama : Abdur Rouf
   b. NIM : 170441100037
   c. Prodi : S1 Sistem Informasi, FT UTM
   d. Asal : Jepara, Jawa Tengah
   e. HP/WA : 089614082087
   f. Email : abdurrouf211214@gmail.com

2. Riwayat Pendidikan
   a. SD N 1 Mindahan (2005-2011)
   b. MTs Nurul Muslim (2011-2014)
   c. MAN 1 Jepara (2014-2017)

SEKIAN

TERIMA KASIH