## IF2240 - Basis Data

# Milestone III

## Desain Basis Data Relasional dan Implementasi Business Rule


**Kelompok 2 - K3**
**Anggota Kelompok :**

| | |
|---|---|
| Afnan Edsa Ramadhan | 13521011 |
| Bernardus Willson | 13521021 |
| Kenny Benaya Nathan | 13521023 |
| Muhammad Haidar Akita Tresnadi | 13521025 |
| M Zulfiansyah Bayu Pratama | 13521028 |
| Jauza Lathifah Annassalafi | 13521030 |

**Program Studi Teknik Informatika**
**Sekolah Teknik Elektro dan Informatika**
**Institut Teknologi Bandung**

**2023**

# DAFTAR ISI

# 1. RELATIONAL DATABASE DESIGN

## 1.1. SKEMA BASIS DATA AWAL SEBELUM NORMALISASI.

```
pasien = (idPasien, namaDepanPasien, namaBelakangPasien, tanggalLahir,
      jenisKelamin, nomorTelepon, email, jalan, kelurahan, kecamatan,
      kabupatenKota, provinsi, namaKontakDarurat, nomorKontakDarurat)
tenagaMedis = (idTenagaMedis, namaDepan, namaBelakang, tanggalLahir,
      jenisKelamin, nomorTelepon, email, jalan, kelurahan, kecamatan,
      kabupatenKota, provinsi, nomorLisensi, tipe)
perusahaanAsuransi = (idPerusahaan, namaKontak, nomorTelepon)
rekamMedis = (idRekamMedis, tanggalMasuk, tanggalKeluar, totalBiaya,
      diagnosis, idPasien)
dokter = (idDokter, spesialisasi)
perawat = (idPerawat)
asuransi = (idAsuransi, tanggalKlaim, nilaiKlaim, kondisi,
      idPerusahaan, idRekamMedis)
prosedur = (idProsedur, namaProsedur, tanggalProsedur, idRekamMedis)
obat = (idObat, namaObat, dosis, tanggalMulai, tanggalSelesai,
      idRekamMedis)
hasilLab = (idHasilLab, hasil, tanggalTes, idRekamMedis)
ruangRawat = (idRuang, lantai, kelas, kapasistas)
melakukan = (idTenagaMedis, idProsedur)
menangani = (idDokter, idPerawat, idRekamMedis)
membantu = (idDokter, idPerawat)
```

## 1.2. IDENTIFIKASI FUNCTIONAL DEPENDENCIES (FD) DAN JENIS BENTUK NORMAL DARI SETIAP RELASI.

### 1.2.1. RELASI pasien

```
pasien = (idPasien, namaDepanPasien, namaBelakangPasien,
      tanggalLahir, jenisKelamin, nomorTelepon, email, jalan,
```

kelurahan, kecamatan, kabupatenKota, provinsi, namaKontakDarurat, nomorKontakDarurat)

FD = {idPasien → namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, namaKontakDarurat, nomerKontakDarurat}

{kelurahan → kecamatan}

{kecamatan → kabupatenKota}

{kabupatenKota → provinsi}

Candidate Key = idTenagaMedis

FD tersebut mengindikasikan bahwa idPasien adalah kunci atau kandidat kunci untuk relasi pasien . Setiap nilai dalam atribut idTenagaMedis secara fungsional menentukan nilai dalam atribut namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, namaKontakDarurat, nomerKontakDarurat. Artinya, untuk setiap idPasien, hanya ada satu dari semua atribut yang sesuai. Kelurahan, kecamatan serta kabupatenKota juga termasuk kandidat kunci untuk relasi pasien dan menentukan nilai dari masing masing atributnya.

Relasi pasien sebelum dinormalisasi berada pada bentuk 2NF karena functional dependencies pada relasi pasien bergantung pada superkey, yaitu idPasien.

R1

idTenagaMedis = (namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, namaKontakDarurat, nomerKontakDarurat)

FD1

idTenagaMedis → namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, namaKontakDarurat, nomerKontakDarurat

R2

kelurahan = (kecamatan)

FD2

kelurahan → kecamatan

R3

kecamatan = (kabupaten)

FD4

kecamatan → kabupaten

R4

kabupaten = (provinsi)

FD4

kabupaten → provinsi

Skema akhir

idTenagaMedis = (idTenagaMedis, namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, namaKontakDarurat, nomerKontakDarurat)

kelurahan = (kelurahan, kecamatan)

kecamatan = (kecamatan, kabupatenKota)

kabupatenKota = (kabupatenKota, provinsi)

Setelah dilakukan normalisasi dengan memecah relasi menjadi 4 bagian, didapat bentuk normal BCNF.

### 1.2.2. RELASI tenagaMedis

```
tenagaMedis = (idTenagaMedis, namaDepan, namaBelakang,
      tanggalLahir, jenisKelamin, nomorTelepon, email, jalan,
      kelurahan, kecamatan, kabupatenKota, provinsi,
      nomorLisensi, tipe)
```

FD = {idTenagaMedis → namaDepan, namaBelakang, tanggalLahir,

jenisKelamin, nomorTelepon, email, jalan, kelurahan, nomorLisensi, tipe}

{kelurahan → kecamatan}

{kecamatan → kabupatenKota}

{kabupatenKota → provinsi}

Candidate Key = idTenagaMedis

FD tersebut mengindikasikan bahwa idTenagaMedis adalah kunci atau kandidat kunci untuk relasi pasien . Setiap nilai dalam atribut idTenagaMedis secara fungsional menentukan nilai dalam atribut namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, namaKontakDarurat, nomerKontakDarurat. Artinya, untuk setiap idTenagaMedis, hanya ada satu dari semua atribut yang sesuai. Kelurahan, kecamatan serta kabupatenKota juga termasuk kandidat kunci untuk relasi pasien dan menentukan nilai dari masing masing atributnya.

Relasi pasien sebelum dinormalisasi berada pada bentuk 2NF karena functional dependencies pada relasi pasien bergantung pada superkey, yaitu idTenagaMedis.

R1

idTenagaMedis = (namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, nomorLisensi, tipe)

FD1

idTenagaMedis → namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, nomorLisensi, tipe

R2

kelurahan = (kecamatan)

FD2

kelurahan → kecamatan

R3

kecamatan = (kabupaten)

FD4

kecamatan → kabupaten

R4

kabupaten = (provinsi)

FD4

kabupaten → provinsi

Skema akhir

idTenagaMedis = (idTenagaMedis, namaDepan, namaBelakang, tanggalLahir, jenisKelamin, nomorTelepon, email, jalan, kelurahan, nomorLisensi, tipe)

kelurahan = (kelurahan, kecamatan)

kecamatan = (kecamatan, kabupatenKota)

kabupatenKota = (kabupatenKota, provinsi)

Setelah dilakukan normalisasi dengan memecah relasi menjadi 4 bagian, didapat bentuk normal BCNF.

### 1.2.3. RELASI perusahaanAsuransi

```
perusahaanAsuransi = (idPerusahaan, namaKontak, nomorTelepon)
```

FD = {idPerusahaan → namaKontak, nomorTelpon}

Candidate Key = idPerusahaan

FD tersebut mengindikasikan bahwa idPerusahaan adalah kunci atau kandidat kunci untuk relasi PerusahaanAsuransi. Setiap nilai dalam

atribut idPerusahaan secara fungsional menentukan nilai dalam atribut namaKontak dan nomorTelepon. Artinya, untuk setiap idPerusahaan, hanya ada satu namaKontak dan nomorTelepon yang sesuai.

Relasi perusahaanAsuransi berada pada bentuk BCNF karena functional dependencies pada relasi perusahaanAsuransi bergantung pada superkey, yaitu idPerusahaan.

### 1.2.4. RELASI rekamMedis

```
rekamMedis = (idRekamMedis, tanggalMasuk, tanggalKeluar,
              totalBiaya, diagnosis, idPasien)
```

FD = {idRekamMedis → tanggalMasuk, tanggalKeluar, totalBiaya, diagnosis, idPasien}

Candidate Key = idRekamMedis

FD tersebut mengindikasikan bahwa idRekamMedis adalah kandidat kunci untuk relasi "rekamMedis". Setiap nilai dalam atribut idRekamMedis secara fungsional menentukan nilai-nilai dalam atribut tanggalMasuk, tanggalKeluar, totalBiaya, diagnosis, dan idPasien.

Relasi rekamMedis berada pada bentuk BCNF karena functional dependencies pada relasi rekamMedis bergantung pada superkey, yaitu idrekamMedis.

### 1.2.5. RELASI dokter

```
dokter = (idDokter, spesialisasi)
```

FD = {idDokter → spesialis}

Candidate Key = idDokter

FD tersebut mengindikasikan bahwa idDokter adalah kandidat kunci untuk relasi dokter. Setiap nilai dalam atribut idDokter secara

fungsional menentukan nilai dalam atribut spesialis. Artinya, untuk setiap idDokter, hanya ada satu spesialis yang sesuai.

Relasi dokter berada pada bentuk BCNF karena functional dependencies pada relasi dokter bergantung pada superkey, yaitu idDokter.

### 1.2.6. RELASI perawat

```
perawat = (idPerawat)
```

Relasi perawat tidak memiliki functional dependencies sehingga tidak dapat ditentukan jenis bentuk normalisasinya.

### 1.2.7. RELASI asuransi

```
asuransi = (idAsuransi, tanggalKlaim, nilaiKlaim, kondisi,
            idPerusahaan, idRekamMedis)
```

FD = {idAsuransi → tanggalKlaim, nilaiKlaim, kondisi, idPerusahaan, idRekamMedis}

Candidate Key = idAsuransi

FD tersebut mengindikasikan bahwa idAsuransi adalah kandidat kunci untuk relasi asuransi. Setiap nilai dalam atribut idAsuransi secara fungsional menentukan nilai dalam atribut tanggalKlaim, nilaiKlaim, kondisi, idPerusahaan, dan idRekamMedis. Artinya, untuk setiap idAsuransi, hanya ada satu tanggalKlaim, nilaiKlaim, kondisi, idPerusahaan, dan idRekamMedis yang sesuai.

Relasi asuransi berada pada bentuk BCNF karena functional dependencies pada relasi asuransi bergantung pada superkey, yaitu idAsuransi.

### 1.2.8. RELASI prosedur

```
prosedur = (idProsedur, namaProsedur, tanggalProsedur,
            idRekamMedis)
```

FD = {idProsedur → namaProsedur, tanggalProsedur, idRekamMedis}

Candidate Key = idProsedur

FD tersebut mengindikasikan bahwa idProsedur adalah kandidat kunci untuk relasi prosedur. Setiap nilai dalam atribut idProsedur

secara fungsional menentukan nilai dalam atribut namaProsedur, tanggalProsedur, dan idRekamMedis. Artinya, untuk setiap idProsedur, hanya ada satu namaProsedur, tanggalProsedur, dan idRekamMedis yang sesuai.

Relasi prosedur berada pada bentuk BCNF karena functional dependencies pada relasi prosedur bergantung pada superkey, yaitu idProsedur.

### 1.2.9. RELASI obat

```
obat = (idObat, namaObat, dosis, tanggalMulai, tanggalSelesai,
       idRekamMedis)
```

FD = {idObat → namaObat, dosis, tanggalMulai, tanggalSelesai, idRekamMedis}

Candidate Key = idObat

FD tersebut mengindikasikan bahwa idObat adalah kandidat kunci untuk relasi obat. Setiap nilai dalam atribut idObat secara fungsional menentukan nilai dalam atribut namaObat, dosis, tanggalMulai, tanggalSelesai, dan idRekamMedis. Artinya, untuk setiap idObat, hanya ada satu namaObat, dosis, tanggalMulai, tanggalSelesai, dan idRekamMedis yang sesuai.

Relasi obat berada pada bentuk BCNF karena functional dependencies pada relasi obatbergantung pada superkey, yaitu idObat.

### 1.2.10. RELASI hasilLab

```
hasilLab = (idHasilLab, hasil, tanggalTes, idRekamMedis)
```

FD = {idHasilLab → hasil, tanggalTes, idRekamMedis}
Candidate Key = idHasilLab

FD tersebut mengindikasikan bahwa idHasilLab adalah kandidat kunci untuk relasi hasilLab. Setiap nilai dalam atribut idHasilLab secara fungsional menentukan nilai dalam atribut hasil, tanggalTes, dan idRekamMedis. Artinya, untuk setiap idHasilLab, hanya ada satu hasil, tanggalTes, dan idRekamMedis yang sesuai.

Relasi hasilLab berada pada bentuk BCNF karena functional dependencies pada relasi hasilLab bergantung pada superkey, yaitu idHasilLab.

### 1.2.11. RELASI melakukan

```
melakukan = (idTenagaMedis, idProsedur)
```

Relasi melakukan tidak memiliki functional dependencies sehingga tidak dapat ditentukan jenis bentuk normalisasinya.

### 1.2.12. RELASI menangani

```
menangani = (idDokter, idPerawat, idRekamMedis)
```

Relasi menangani tidak memiliki functional dependencies sehingga tidak dapat ditentukan jenis bentuk normalisasinya.

### 1.2.13. RELASI membantu

```
membantu = (idDokter, idPerawat)
```

Relasi membantu tidak memiliki functional dependencies sehingga tidak dapat ditentukan jenis bentuk normalisasinya.

### 1.2.14. RELASI ruangRawat

```
ruangRawat= (idRuang, lantai, kelas, kapasitas)
```

FD = {idRuang → lantai, kelas, kapasitas}

Candidate Key = {idRuang}

FD tersebut mengindikasikan bahwa idRuang adalah kandidat kunci untuk relasi ruangRawat. Setiap nilai dalam atribut idRuang secara fungsional menentukan nilai dalam atribut lantai, kelas, dan kapasitas. Artinya, untuk setiap idRuang, hanya ada satu lantai, kelas, dan kapasitas yang sesuai.

Relasi ruangRawat berada pada bentuk BCNF karena functional dependencies pada relasi ruangRawat bergantung pada superkey, yaitu idRuang.

## 1.3. SKEMA BASIS DATA HASIL NORMALISASI

```
pasien = (idPasien, namaDepanPasien, namaBelakangPasien, tanggalLahir,
      jenisKelamin, nomorTelepon, email, jalan, kelurahan, kecamatan,
      kabupatenKota, provinsi, namaKontakDarurat, nomorKontakDarurat)
tenagaMedis = (idTenagaMedis, namaDepan, namaBelakang, tanggalLahir,
      jenisKelamin, nomorTelepon, email, jalan, kelurahan, kecamatan,
      kabupatenKota, provinsi, nomorLisensi, tipe)
daftarKelurahan = (kelurahan, kecamatan)
daftarKecamatan = (kecamatan, kabupatenKota)
daftarKabupatenKota = (kabupatenKota, provinsi)
perusahaanAsuransi = (idPerusahaan, namaKontak, nomorTelepon)
rekamMedis = (idRekamMedis, tanggalMasuk, tanggalKeluar, totalBiaya,
```

```
        diagnosis, idPasien)
dokter = (idDokter, spesialisasi)
perawat = (idPerawat)
asuransi    =    (idAsuransi,    tanggalKlaim,    nilaiKlaim,    kondisi,
        idPerusahaan, idRekamMedis)
prosedur = (idProsedur, namaProsedur, tanggalProsedur, idRekamMedis)
obat   =   (idObat,   namaObat,   dosis,   tanggalMulai,   tanggalSelesai,
        idRekamMedis)
hasilLab = (idHasilLab, hasil, tanggalTes, idRekamMedis)
ruangRawat = (idRuang, lantai, kelas, kapasistas)
melakukan = (idTenagaMedis, idProsedur)
menangani = (idDokter, idPerawat, idRekamMedis)
membantu = (idDokter, idPerawat)
```

## 2. BUSINESS RULE

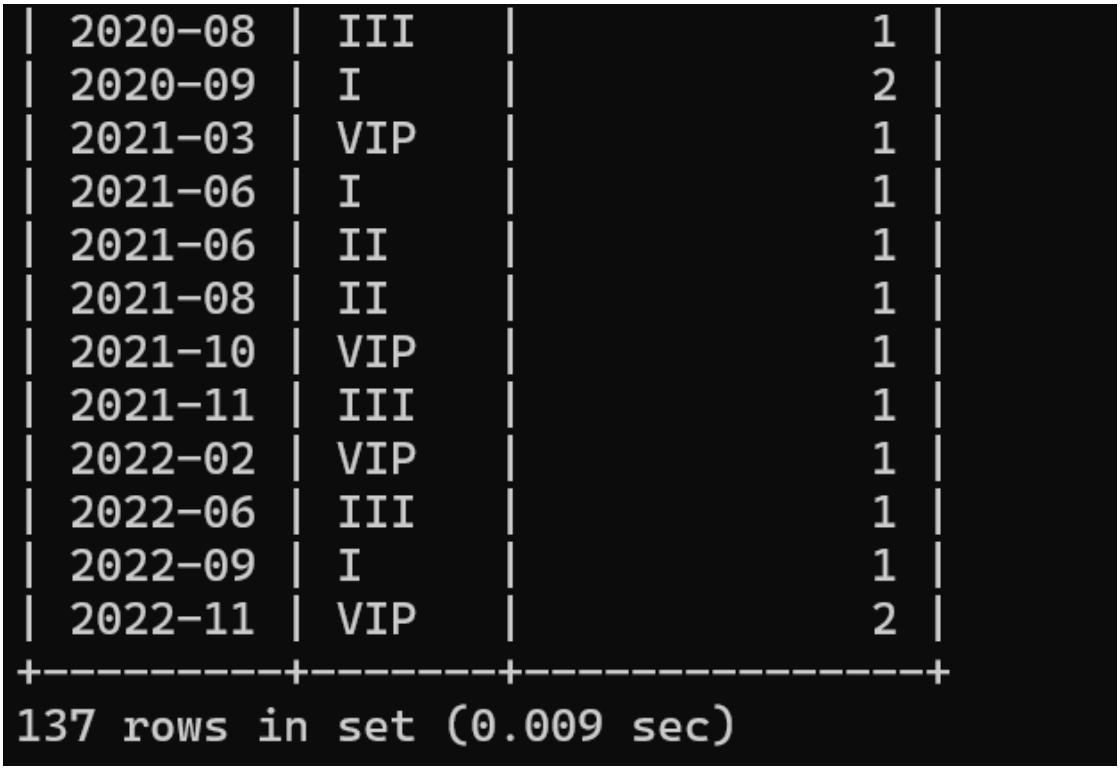
| | |
|---|---|
| 1. | **INSIGHT** |
| | Buatlah sebuah view untuk menampilkan jumlah pasien yang dirawat di rumah sakit pada setiap bulan, dikelompokkan berdasarkan kelas ruang rawatnya. |
| | **QUERY** |
| | CREATE VIEW jumlah_pasien_per_bulan AS<br>SELECT<br>  DATE_FORMAT(tanggalMasuk, '%Y-%m') AS bulan,<br>  kelas,<br>  COUNT(*) AS jumlah_pasien<br>FROM<br>  rekamMedis NATURAL JOIN ruangRawat<br>GROUP BY<br>  bulan, |

kelas;

**SCREENSHOT HASIL QUERY**

```
MariaDB [mils3]> CREATE VIEW jumlah_pasien_per_bulan AS
    -> SELECT
    ->     DATE_FORMAT(tanggalMasuk, '%Y-%m') AS bulan,
    ->     kelas,
    ->     COUNT(*) AS jumlah_pasien
    -> FROM
    ->     rekamMedis NATURAL JOIN ruangRawat
    -> GROUP BY
    ->     bulan,
    ->     kelas;
Query OK, 0 rows affected (0.007 sec)
```

```
MariaDB [mils3]> SELECT * FROM jumlah_pasien_per_bulan;
+---------+-------+---------------+
| bulan   | kelas | jumlah_pasien |
+---------+-------+---------------+
| 2013-04 | II    |             2 |
| 2013-05 | I     |             1 |
| 2013-05 | III   |             3 |
| 2013-05 | VIP   |             1 |
| 2013-06 | I     |             1 |
| 2013-06 | VIP   |             2 |
| 2013-07 | I     |             1 |
| 2013-07 | II    |             1 |
| 2013-07 | VIP   |             2 |
| 2013-08 | I     |             1 |
| 2013-08 | II    |             1 |
| 2013-09 | II    |             1 |
| 2013-09 | III   |             1 |
| 2013-09 | VIP   |             1 |
| 2013-10 | I     |             1 |
```



```
| 2020-08 | III  |              1 |
| 2020-09 | I    |              2 |
| 2021-03 | VIP  |              1 |
| 2021-06 | I    |              1 |
| 2021-06 | II   |              1 |
| 2021-08 | II   |              1 |
| 2021-10 | VIP  |              1 |
| 2021-11 | III  |              1 |
| 2022-02 | VIP  |              1 |
| 2022-06 | III  |              1 |
| 2022-09 | I    |              1 |
| 2022-11 | VIP  |              2 |
+---------+------+----------------+
137 rows in set (0.009 sec)
```

| 2. | **INSIGHT** |
|---|---|
| | Buatlah prosedur untuk meng-update total biaya pada rekaman medis berdasarkan aktivitas medis terbaru yang ditambahkan. Aktivitas medis dapat berupa pengobatan, tes laboratorium, ataupun prosedur. Tambahkan atribut biaya pada aktivitas medis apabila diperlukan. Kemudian, buatlah sebuah trigger untuk memastikan data aktivitas medis yang dimasukkan valid. Valid yang dimaksud berarti tanggal pelaksanaannya berada di antara tanggal masuk dan tanggal keluar yang tercatat pada rekaman medis. Jika aktivitas medis tersebut valid, panggil prosedur yang sudah dibuat sebelumnya untuk menjaga konsistensi data rekaman medis! |
| | **QUERY** |
| | DELIMITER //<br>-- cek tanggal prosedur<br>CREATE TRIGGER check_tanggalProsedur |

```
BEFORE INSERT ON prosedur
FOR EACH ROW
BEGIN
  DECLARE tglMasuk DATE;
  DECLARE tglKeluar DATE;

  -- Get the tanggalMasuk and tanggalKeluar for the corresponding rekamMedis
row
  SELECT tanggalMasuk, tanggalKeluar
  INTO tglMasuk, tglKeluar
  FROM rekamMedis
  WHERE idRekamMedis = NEW.idRekamMedis;

  -- Check that the tanggalProsedur is between the tanggalMasuk and
tanggalKeluar
  IF (NEW.tanggalProsedur < tglMasuk OR NEW.tanggalProsedur > tglKeluar) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'tanggalProsedur is not within
the date range of the corresponding rekamMedis row';
  END IF;
END
//
DELIMITER ;


DELIMITER //
-- cek tanggal tes
CREATE TRIGGER check_tanggalTes
BEFORE INSERT ON hasilLab
FOR EACH ROW
BEGIN

  -- Get the tanggalMasuk and tanggalKeluar for the corresponding rekamMedis
row
  SELECT tanggalMasuk, tanggalKeluar
  FROM rekamMedis
  WHERE idRekamMedis = NEW.idRekamMedis;

  -- Check that the tanggalTes is between the tanggalMasuk and tanggalKeluar
  IF (NEW.tanggalTes < tglMasuk OR NEW.tanggalTes > tglKeluar) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'tanggalTes is not within the
date range of the corresponding rekamMedis row';
  END IF;
```

```
END
//
DELIMITER ;

DELIMITER //
CREATE PROCEDURE update_total_biaya(IN rekam_id INT, IN new_biaya BIGINT)
BEGIN
  UPDATE rekamMedis
  SET totalBiaya = totalBiaya + new_biaya
  WHERE idRekamMedis = rekam_id;
END;
//
DELIMITER ;
```

**SCREENSHOT HASIL QUERY**

```
MariaDB [mils3]> DELIMITER //
MariaDB [mils3]> -- cek tanggal prosedur
MariaDB [mils3]> CREATE TRIGGER check_tanggalProsedur
    -> BEFORE INSERT ON prosedur
    -> FOR EACH ROW
    -> BEGIN
    ->   DECLARE tglMasuk DATE;
    ->   DECLARE tglKeluar DATE;
    ->
    ->   -- Get the tanggalMasuk and tanggalKeluar for the corresponding rekamMedis row
    ->   SELECT tanggalMasuk, tanggalKeluar
    ->   INTO tglMasuk, tglKeluar
    ->   FROM rekamMedis
    ->   WHERE idRekamMedis = NEW.idRekamMedis;
    ->
    ->   -- Check that the tanggalProsedur is between the tanggalMasuk and tanggalKeluar
    ->   IF (NEW.tanggalProsedur < tglMasuk OR NEW.tanggalProsedur > tglKeluar) THEN
    ->     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'tanggalProsedur is not within the date range of the corre
sponding rekamMedis row';
    ->   END IF;
    -> END
    -> //
Query OK, 0 rows affected (0.030 sec)
```

```
MariaDB [mils3]> DELIMITER //
MariaDB [mils3]> -- cek tanggal pengobatan
MariaDB [mils3]> CREATE TRIGGER check_tanggalPengobatan
    -> BEFORE INSERT ON obat
    -> FOR EACH ROW
    -> BEGIN
    ->   DECLARE tglMasuk DATE;
    ->   DECLARE tglKeluar DATE;
    ->
    ->   -- Get the tanggalMasuk and tanggalKeluar for the corresponding rekamMedis row
    ->   SELECT tanggalMasuk, tanggalKeluar
    ->   INTO tglMasuk, tglKeluar
    ->   FROM rekamMedis
    ->   WHERE idRekamMedis = NEW.idRekamMedis;
    ->
    ->   -- Check that the tanggalMulai and tanggalSelesai is between the tanggalMasuk and tanggalKeluar
    ->   IF (NEW.tanggalMulai < tglMasuk OR
    ->     NEW.tanggalMulai > tglKeluar OR
    ->     NEW.tanggalSelesai < tglMasuk) THEN
    ->     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'tanggalMulai is not within the date range of the correspo
nding rekamMedis row';
    ->   END IF;
    -> END
    -> //
Query OK, 0 rows affected (0.015 sec)

MariaDB [mils3]> DELIMITER ;
```

```
MariaDB [mils3]> DELIMITER //
MariaDB [mils3]> -- cek tanggal tes
MariaDB [mils3]> CREATE TRIGGER check_tanggalTes
    -> BEFORE INSERT ON hasilLab
    -> FOR EACH ROW
    -> BEGIN
    ->   DECLARE tglMasuk DATE;
    ->   DECLARE tglKeluar DATE;
    ->   -- Get the tanggalMasuk and tanggalKeluar for the corresponding rekamMedis row
    ->   SELECT tanggalMasuk, tanggalKeluar
    ->   INTO tglMasuk, tglKeluar
    ->   FROM rekamMedis
    ->   WHERE idRekamMedis = NEW.idRekamMedis;
    ->
    ->   -- Check that the tanggalTes is between the tanggalMasuk and tanggalKeluar
    ->   IF (NEW.tanggalTes < tglMasuk OR NEW.tanggalTes > tglKeluar) THEN
    ->     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'tanggalTes is not within the date range of the correspond
ing rekamMedis row';
    ->   END IF;
    -> END
    -> //
Query OK, 0 rows affected (0.015 sec)
```

3.

**INSIGHT**

Pastikan maksimal hanya satu klaim asuransi yang memiliki status "diterima" dalam satu rekaman medis. Pastikan juga nominal biaya yang diajukan pada klaim asuransi tidak melebihi total biaya pada rekaman medis!

**QUERY**

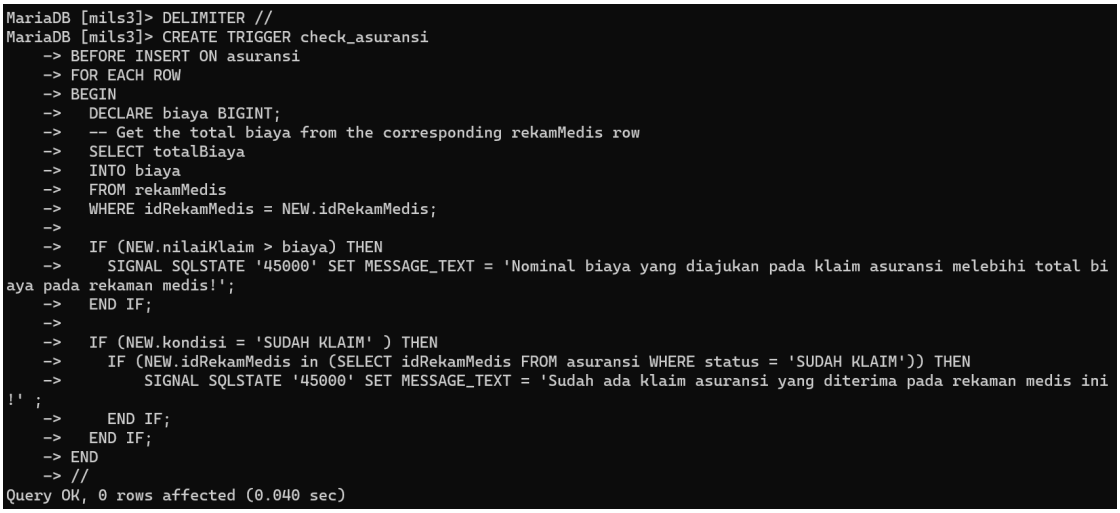
```
DELIMITER //
CREATE TRIGGER check_asuransi
BEFORE INSERT ON asuransi
FOR EACH ROW
BEGIN
  DECLARE biaya BIGINT;
  -- Get the total biaya from the corresponding rekamMedis row
  SELECT totalBiaya
  INTO biaya
  FROM rekamMedis
  WHERE idRekamMedis = NEW.idRekamMedis;

  IF (NEW.nilaiKlaim > biaya) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Nominal biaya yang diajukan
pada klaim asuransi melebihi total biaya pada rekaman medis!';
  END IF;

  IF (NEW.kondisi = 'DITERIMA' ) THEN
    IF (NEW.idRekamMedis in (SELECT idRekamMedis FROM asuransi WHERE
kondisi= 'DITERIMA')) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Sudah ada klaim asuransi
```

```
yang diterima pada rekaman medis ini!' ;
   END IF;
  END IF;
END
//
DELIMITER ;
```

**SCREENSHOT HASIL QUERY**



```
MariaDB [mils3]> DELIMITER //
MariaDB [mils3]> CREATE TRIGGER check_asuransi
    -> BEFORE INSERT ON asuransi
    -> FOR EACH ROW
    -> BEGIN
    ->   DECLARE biaya BIGINT;
    ->   -- Get the total biaya from the corresponding rekamMedis row
    ->   SELECT totalBiaya
    ->   INTO biaya
    ->   FROM rekamMedis
    ->   WHERE idRekamMedis = NEW.idRekamMedis;
    ->
    ->   IF (NEW.nilaiKlaim > biaya) THEN
    ->     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Nominal biaya yang diajukan pada klaim asuransi melebihi total bi
aya pada rekaman medis!';
    ->   END IF;
    ->
    ->   IF (NEW.kondisi = 'SUDAH KLAIM' ) THEN
    ->     IF (NEW.idRekamMedis in (SELECT idRekamMedis FROM asuransi WHERE status = 'SUDAH KLAIM')) THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Sudah ada klaim asuransi yang diterima pada rekaman medis ini
!' ;
    ->     END IF;
    ->   END IF;
    -> END
    -> //
Query OK, 0 rows affected (0.040 sec)
```

## 3. KESIMPULAN

Dalam merancang sebuah basis data, penting untuk memperhatikan keterkaitan fungsional antar atribut. Dengan memperhatikan keterkaitan fungsional ini, kita dapat mengevaluasi tingkat normalisasi dari tabel-tabel yang ada dalam basis data. Dengan desain yang baik, kita dapat menciptakan skema data yang optimal dan mencapai tingkat normalisasi yang paling tinggi untuk setiap relasi dalam basis data.

Setelah menganalisis desain basis data yang telah dibuat, kami menyimpulkan bahwa ada beberapa relasi dalam basis data telah mencapai bentuk BCNF. Untuk relasi yang belum mencapai bentuk BCNF, telah dilakukan normalisasi sehingga dapat mencapai bentuk tertinggi, sehingga dapat menunjukkan bahwa desain basis data memenuhi kriteria normalisasi yang baik. Selain itu, semua kunci calon dan ketergantungan fungsional telah terdefinisi dengan jelas pada setiap relasi.

Penerapan aturan bisnis (business rules) sangatlah penting dalam proses desain basis data. Salah satu bentuk implementasi dari aturan bisnis adalah untuk membentuk tampilan (view) dan menampilkan data dengan cara yang spesifik. Hal ini memiliki manfaat yang signifikan dalam memastikan integritas dan keamanan sistem, serta mengoptimalkan kinerja sistem secara keseluruhan.

# 4. REFERENSI

Bahan Kuliah IF2240 Basis Data Tahun 2023