1. Mengapa desain database diperlukan?

Desain Database diperlukan menyimpan data data krusial dan memudahkan untuk melakukan pengelolaan data ,Memungkinkan penyimpanan data dalam format yang terorganisir , bisa juga mendapat data secara efisien karena sudah tertata

2. Kesalahan umum dalam desain database dan dampaknya ?

1. Kesalahan umum bisanya terjadi saat pembuat/programmer menyimpan data secara redundant yang menyebabkan pemborosan ruang penyimpanan database tersebut.
2. Mengabaikan relasi antar table,Yang menyebabkan keterkaitan antar data
3. Tidak adanya backup dan recovery plan, menghindari adanya human error dan berjaga jaga untuk membuat backup.
4. Kurangnya Indexing atau Terlalu Banyak Indexing Dampaknya tanpa index bisa membuat memperlambat saat pencarian,sebaliknya jika terlalu banyak memperlambat operasi INSERT, UPDATE, dan DELETE.

3. Konsep normalisasi (1NF, 2NF, 3NF, BCNF). ?

1. **First Normal Form (1NF):**

**Syarat:**

Setiap kolom harus memiliki **nilai atomik** (tidak boleh berisi daftar atau array).

Setiap baris harus unik dengan adanya **primary key**.

**Contoh Tidak 1NF:**

| **ID\_Pelanggan** | **Nama** | **No\_Telepon** |
| --- | --- | --- |
| 1 | Ali | 08123, 08234 |
| 2 | Budi | 08567 |

**Perbaikan ke 1NF:**

| **ID\_Pelanggan** | **Nama** | **No\_Telepon** |
| --- | --- | --- |
| 1 | Ali | 08123 |
| 1 | Ali | 08234 |
| 2 | Budi | 08567 |

CONTOH KODE UNTUK 1NF:

CREATE TABLE `USER\_1NF` (

    ID\_PELANGGAN INT PRIMARY KEY AUTO\_INCREMENT,

    NAME VARCHAR(50) NOT NULL

)

CREATE TABLE CONTACT\_PERSON (

    ID\_PELANGGAN INT AUTO\_INCREMENT,

    NAME VARCHAR(50) NOT NULL,

    NO\_TELEPON VARCHAR(50) NOT NULL,

    EMAIL VARCHAR(50) NOT NULL,

    FOREIGN KEY (ID\_PELANGGAN) *REFERENCES* `USER\_1NF`(ID\_PELANGGAN)

)

CONTOH CODE TIDAK INF1 :

CREATE TABLE `USER\_NOT\_1NF` (

   ID\_PELANGGAN INT PRIMARY KEY AUTO\_INCREMENT,

    NAME VARCHAR(50) NOT NULL

    NO\_ TELEPON VARCHAR(50) NOT NULL,

    EMAIL VARCHAR(50) NOT NULL,

)

1. **Second Normal Form (2NF):**

**Syarat:**

Sudah **ternomalisasikan** atau **1NF**

Semua kolom **bergantung sepenuhnya pada primary key**, tidak hanya pada sebagian primary key (jika ada primary key gabungan).

**Contoh Tidak 2NF (Primary key: ID\_Pesanan + ID\_Pelanggan):**

| **ID\_Pesanan** | **ID\_Pelanggan** | **Nama** | **Produk** |
| --- | --- | --- | --- |
| 101 | 1 | Ali | Laptop |
| 102 | 2 | Budi | HP |

**Kesalahan:** Nama hanya bergantung pada ID\_Pelanggan, bukan ID\_Pesanan.  
**Perbaikan ke 2NF (Pisahkan tabel pelanggan dan pesanan):**

**Tabel Pelanggan:**

| **ID\_Pelanggan** | **Nama** |
| --- | --- |
| 1 | Ali |
| 2 | Budi |

**Tabel Pesanan:**

| **ID\_Pesanan** | **ID\_Pelanggan** | **Produk** |
| --- | --- | --- |
| 101 | 1 | Laptop |
| 102 | 2 | HP |

Contoh kode untuk 2NF :

CREATE Table pelanggan (

    ID\_PELANGGAN INT PRIMARY KEY AUTO\_INCREMENT,

    NAMA VARCHAR(50) NOT NULL

)

create table pesanan\_pelanggan (

    ID\_PESANAN INT PRIMARY KEY AUTO\_INCREMENT,

    ID\_PELANGGAN INT NOT NULL,

    TANGGAL\_PESAN DATE NOT NULL,

    FOREIGN KEY (ID\_PELANGGAN) *REFERENCES* PELANGGAN(ID\_PELANGGAN)

)

Kode diatas cukup memanggil dari idnya saj tanpa membuat nama pelanggan lagi,supaya tidak berulang ulang

1. **Third Normal Form (3NF)**

**Syarat:**

Sudah dalam **2NF**.

Tidak ada atribut non-prime (bukan primary key) yang bergantung pada atribut non-prime lainnya.

**Contoh Tidak 3NF:**

| **ID\_Pelanggan** | **Nama** | **Kota** | **Kode\_Pos** |
| --- | --- | --- | --- |
| 1 | Ali | Jakarta | 10110 |
| 2 | Budi | Bandung | 40234 |

**Kesalahan:** Kode\_Pos bergantung pada Kota, bukan langsung pada ID\_Pelanggan.  
**Perbaikan ke 3NF:**  
**Tabel Pelanggan:**

| **ID\_Pelanggan** | **Nama** | **Kota** |
| --- | --- | --- |
| 1 | Ali | Jakarta |
| 2 | Budi | Bandung |

**Tabel Kota:**

| **Kota** | **Kode\_Pos** |
| --- | --- |
| Jakarta | 10110 |
| Bandung | 40234 |

**4. Boyce-Codd Normal Form (BCNF) – Mengatasi Masalah 3NF yang Masih Bermasalah**

**Syarat:**

Sudah dalam **3NF**.

Jika ada dependensi fungsional **X → Y**, maka X harus superkey.

**Contoh Tidak BCNF:**

| **ID\_Dosen** | **Mata\_Kuliah** | **Ruangan** |
| --- | --- | --- |
| 1 | Basis Data | A101 |
| 2 | Algoritma | B202 |

**Kesalahan:** Mata\_Kuliah menentukan Ruangan, tetapi ID\_Dosen juga bisa menentukan Mata\_Kuliah.

**Perbaikan ke BCNF:**  
**Tabel Dosen-Mata Kuliah:**

| **ID\_Dosen** | **Mata\_Kuliah** |
| --- | --- |
| 1 | Basis Data |
| 2 | Algoritma |

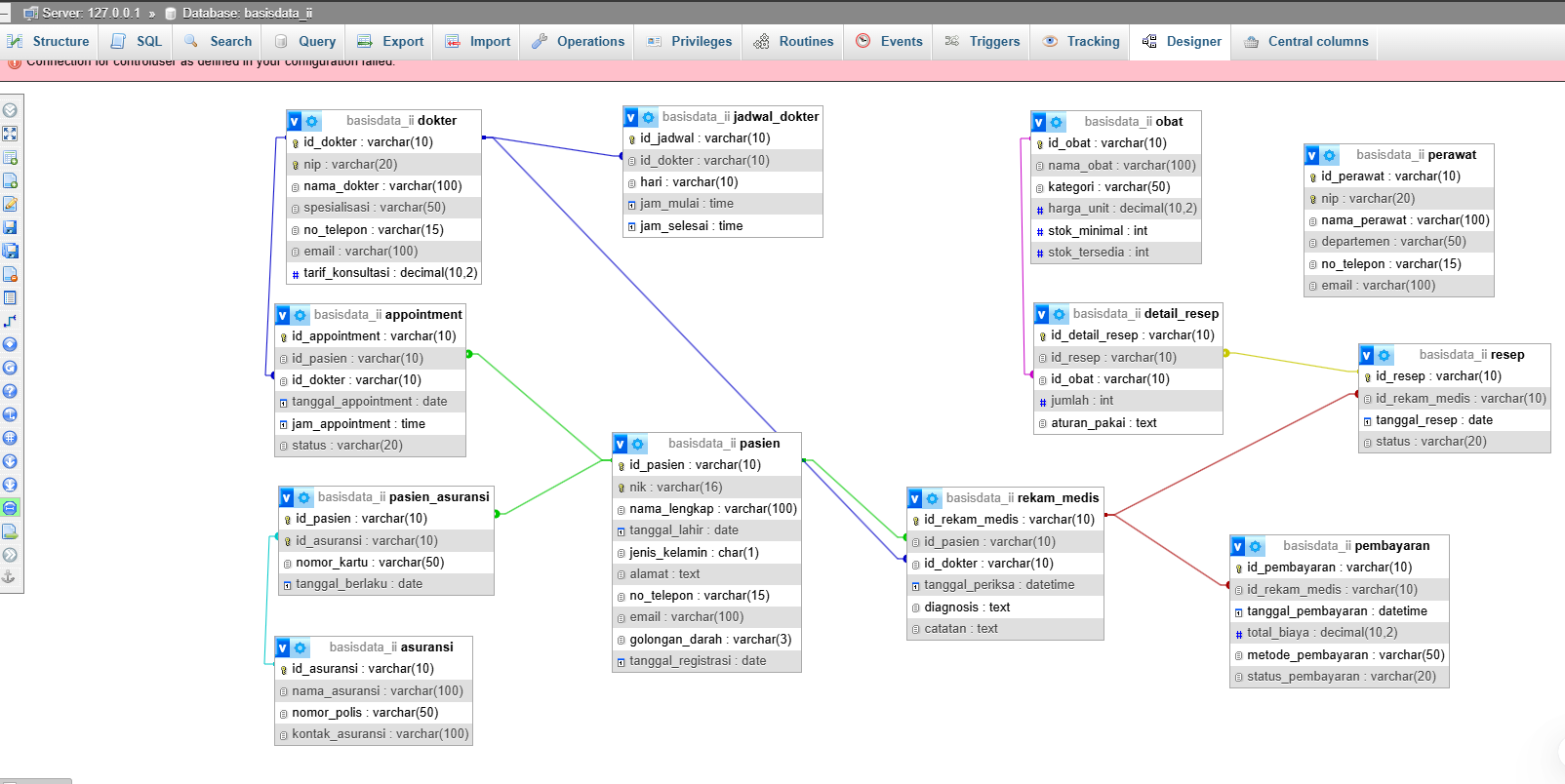
**Tabel Mata Kuliah-Ruangan :**

**RUANGAN MATA KULIAH**

A101 Basis Data

B202 Algoritma

**TUGAS :**



PENJELASAN :

Penjelasan Normalisasi:

1. First Normal Form (1NF):

* Semua tabel memiliki primary key
* Tidak ada repeating groups
* Setiap kolom berisi nilai atomic

1. Second Normal Form (2NF):

* Sudah memenuhi 1NF
* Semua atribut non-key bergantung sepenuhnya pada primary key
* Dipisahkan tabel untuk menghindari partial dependency

1. Third Normal Form (3NF):

* Sudah memenuhi 2NF
* Tidak ada transitive dependency
* Setiap atribut non-key harus bergantung langsung pada primary key

Relasi antar tabel:

* Pasien - Appointment (One to Many)
* Dokter - Appointment (One to Many)
* Pasien - Rekam\_Medis (One to Many)
* Rekam\_Medis - Resep (One to One)
* Resep - Detail\_Resep (One to Many)
* Obat - Detail\_Resep (One to Many)
* Pasien - Pasien\_Asuransi (One to Many)
* Asuransi - Pasien\_Asuransi (One to Many)
* Rekam\_Medis - Pembayaran (One to One)

Desain ini sudah memenuhi kebutuhan:

* Manajemen data pasien
* Penjadwalan dokter dan appointment
* Pencatatan rekam medis
* Pengelolaan resep dan obat
* Manajemen pembayaran dan asuransi
* Pelaporan dan administrasi

Desain ini juga mempertimbangkan:

* Data integrity
* Referential integrity
* Minimal redundancy
* Flexibility untuk pengembangan
* Security dan privacy data pasien