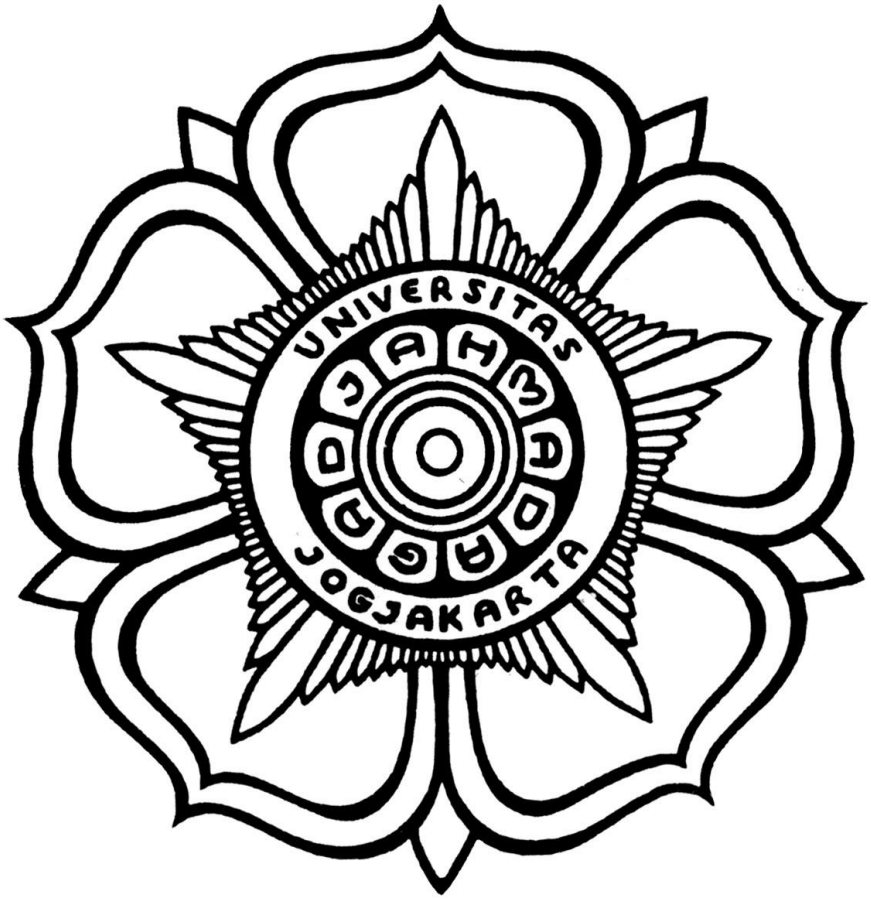
Week 2 – Schema SQL

Clinic Appointment System



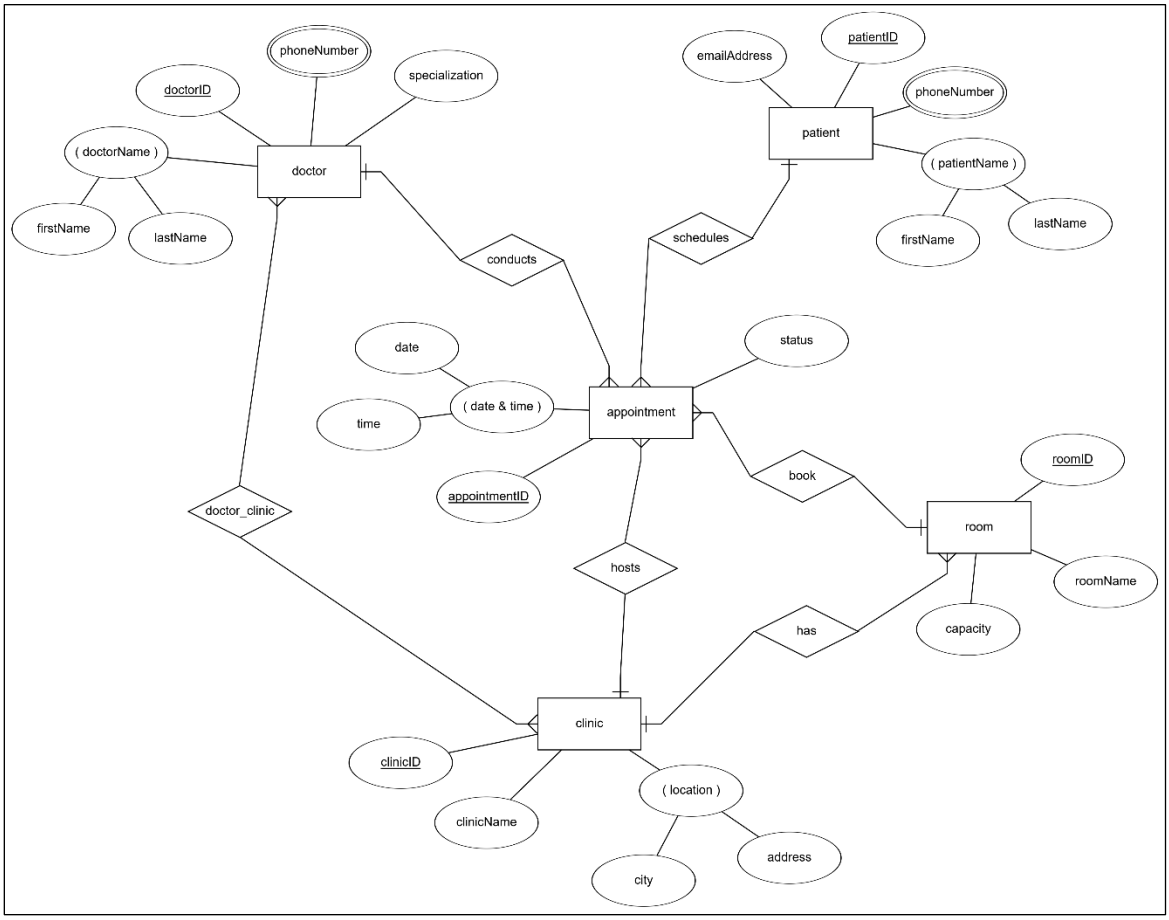
Kelompok 3

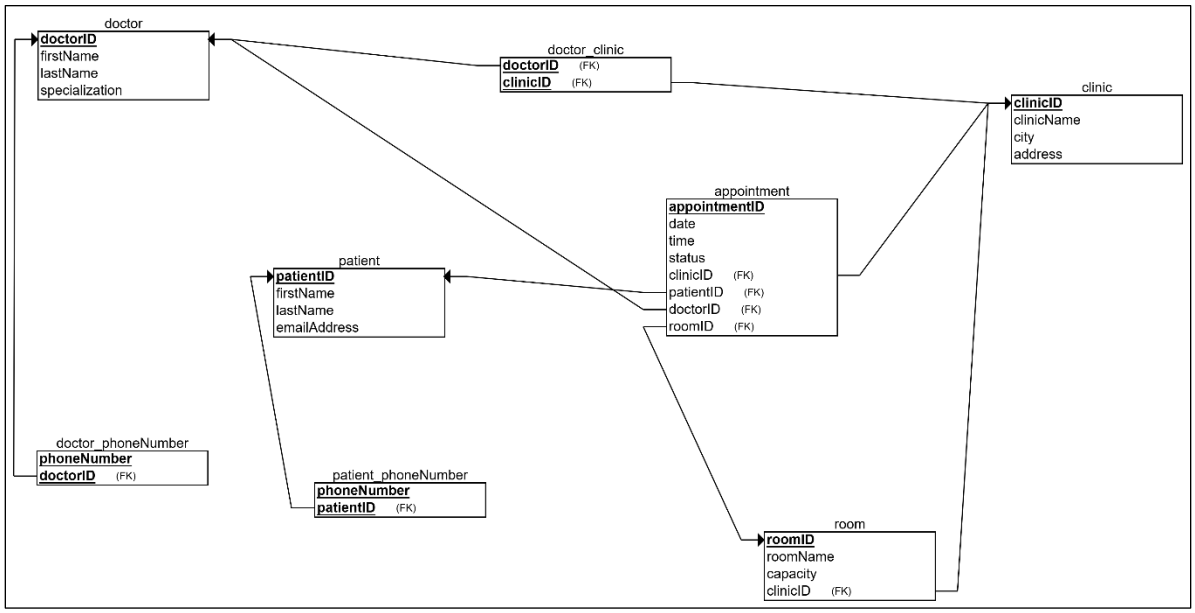
Chelsea Natasja Jesslyne Sembiring (24/543571/PA/23097)

Miera Ardiyanti Farica (24/534436/PA/22665)

KOM A

I. Konversi ERD ke Skema Relasional





Tahap 1: Konversi ERD ke Skema Relasional

Tahap pertama adalah menerjemahkan ERD menjadi skema relasional (daftar tabel dan kolom). Aturan pemetaan standar berikut diterapkan:

- Entitas Kuat (Strong Entities): Setiap entitas kuat (doctor, patient, clinic, room) dipetakan menjadi tabel terpisah.
- Atribut: Setiap atribut menjadi kolom di dalam tabel tersebut (misal: firstName, specialization, clinicName, capacity).
- Atribut Komposit: Atribut komposit seperti doctorName dan patientName dipecah menjadi kolom atomik (firstName, lastName).
- Relasi 1-to-Many: Relasi 1:M diimplementasikan menggunakan *foreign key*.
    - Contoh: Relasi clinic 1--{ room diimplementasikan dengan menambahkan clinicID sebagai *foreign key* di dalam tabel room.
- Relasi Many-to-Many: Relasi M:M diimplementasikan menggunakan tabel persimpangan (*junction table*).
    - Contoh: Relasi doctor }o--o{ clinic diimplementasikan dengan membuat tabel baru, doctor_clinic, yang berisi *foreign key* dari kedua tabel (doctorID dan clinicID).
- Atribut Multivalued: Atribut *multivalued* (seperti phoneNumber untuk dokter dan pasien) diimplementasikan sebagai tabel terpisah (doctor_phoneNumber, patient_phoneNumber).

II. Normalisasi up to 3NF

a. Cek 1NF (First Normal Form):
   - **Aturan:** Setiap sel tabel harus berisi nilai tunggal (atomik), dan tidak boleh ada grup berulang.
   - **Analisis:** telah berhasil memenuhi 1NF.
       - Atribut phoneNumber yang *multivalued* telah dipisah menjadi tabel sendiri: doctor_phoneNumber dan patient_phoneNumber.
       - Atribut komposit (seperti doctorName dari ERD) telah dipecah menjadi firstName dan lastName.
       - Semua kolom lain (seperti date dan time yang dipisah) bersifat atomik.

b. Cek 2NF (Second Normal Form):
   - **Aturan:** Harus dalam 1NF **DAN** tidak boleh ada *partial dependency* (ketergantungan parsial).
   - **Penjelasan:** *Partial dependency* hanya terjadi jika Anda memiliki *composite primary key* (PK dengan banyak kolom). Ini berarti sebuah atribut non-key hanya bergantung pada *sebagian* dari PK tersebut, bukan seluruhnya.
   - **Analisis:** telah berhasil memenuhi 2NF.

- Tabel dengan PK tunggal (`doctor`, `patient`, `clinic`, `room`, `appointment`) otomatis lolos 2NF.
- Kita perlu memeriksa tabel dengan PK komposit:
    - `doctor_clinic (doctorID, clinicID)`: Tidak ada atribut non-key. Lolos 2NF.
    - `doctor_phoneNumber (phoneNumber, doctorID)`: Tidak ada atribut non-key. Lolos 2NF.
    - `patient_phoneNumber (phoneNumber, patientID)`: Tidak ada atribut non-key. Lolos 2NF.

c. Cek 3NF (Third Normal Form):
- **Aturan:** Harus dalam 2NF **DAN** tidak boleh ada *transitive dependency* (ketergantungan transitif).
- **Penjelasan:** Ketergantungan transitif terjadi ketika atribut non-key (A) bergantung pada atribut non-key lainnya (B), yang bergantung pada Primary Key (PK).
    - Bentuknya: **PK -> Atribut_NonKey_B -> Atribut_NonKey_A**
- **Analisis:**
    - doctor, patient, clinic: **Lolos 3NF**. Semua atribut (seperti firstName, city, address) bergantung langsung pada PK-nya (doctorID, clinicID), bukan pada atribut non-key lain.
    - room: **Lolos 3NF**. Atribut roomName dan capacity bergantung pada roomID. clinicID (FK) juga bergantung pada roomID. Tidak ada dependensi transitif.
    - doctor_clinic, ..._phoneNumber: **Lolos 3NF**. (Tidak ada atribut non-key).
    - appointment: **INI MELANGGAR 3NF.**

    - **Penjelasan Pelanggaran 3NF pada Tabel appointment**
        - **Primary Key:** appointmentID
        - **Atribut Non-Key (termasuk FK):** date, time, status, clinicID, patientID, doctorID, roomID

          Perhatikan dependensi (ketergantungan) berikut:
            1. appointmentID menentukan roomID (Benar. Janji temu ini ada di ruangan mana).
            2. roomID menentukan clinicID (Benar. Dilihat dari tabel room, setiap roomID pasti berlokasi di satu clinicID).

          Ini menciptakan **dependensi transitif**: **appointmentID → roomID → clinicID**
            - **Masalahnya:** memiliki clinicID, sebuah atribut non-key, yang bergantung pada roomID, atribut non-key lainnya. Ini menyebabkan **redundansi data**.
            - **Contoh:** Jika ada 100 janji temu di roomID = 101, akan disimpan clinicID = "KLINIK-A" sebanyak 100 kali di

dalam tabel appointment, padahal informasi itu sudah ada di tabel room.

- **Solusi untuk Mencapai 3NF**

  Untuk memperbaiki ini, harus **dihapus kolom yang menyebabkan dependensi transitif** dari tabel appointment.

  Hapus clinicID dari tabel appointment.

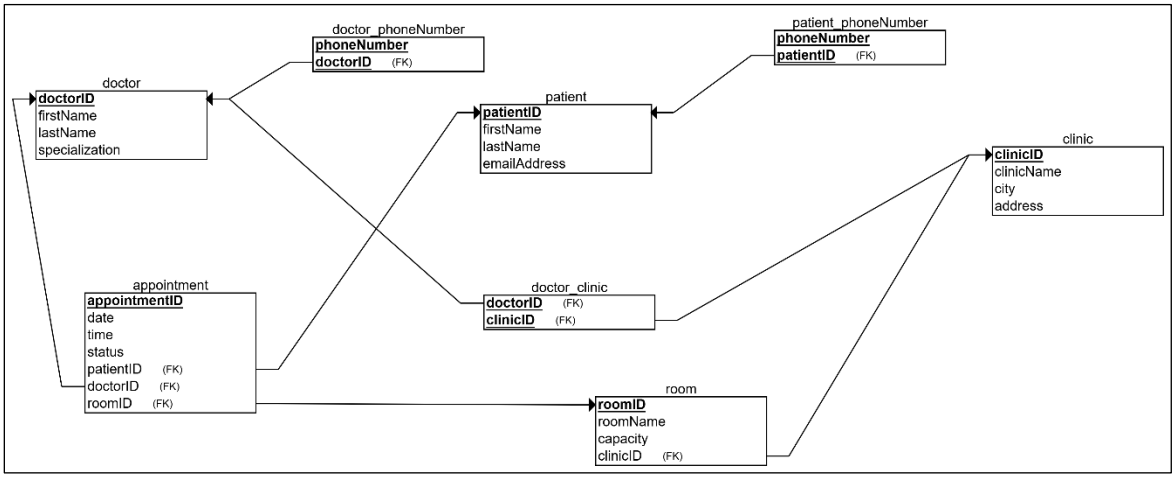
  Mengapa ini berhasil?

  - Informasi clinicID tidak hilang.
  - Jika perlu tahu di klinik mana sebuah janji temu diadakan, dapat ditemukan melalui roomID.

III. Implementasi SQL (DDL)

```
CREATE TABLE doctor
(
  doctorID INT NOT NULL,
  firstName VARCHAR(100) NOT NULL,
  lastName VARCHAR(100) NOT NULL,
  specialization VARCHAR(100) NOT NULL,
  PRIMARY KEY (doctorID)
);

CREATE TABLE patient
(
  patientID INT NOT NULL,
  firstName VARCHAR(100) NOT NULL,
  lastName VARCHAR(100) NOT NULL,
  emailAddress VARCHAR(255) NOT NULL,
  PRIMARY KEY (patientID)
);

CREATE TABLE clinic
(
  clinicID INT NOT NULL,
  clinicName VARCHAR(255) NOT NULL,
  city VARCHAR(100) NOT NULL,
  address VARCHAR(255) NOT NULL,
  PRIMARY KEY (clinicID)
);

CREATE TABLE room
(
  roomID INT NOT NULL,
  roomName VARCHAR(100) NOT NULL,
  capacity INT NOT NULL,
  clinicID INT NOT NULL,
  PRIMARY KEY (roomID),
  FOREIGN KEY (clinicID) REFERENCES clinic(clinicID)
);
```

```
CREATE TABLE doctor_clinic
(
  doctorID INT NOT NULL,
  clinicID INT NOT NULL,
  PRIMARY KEY (doctorID, clinicID),
  FOREIGN KEY (doctorID) REFERENCES doctor(doctorID),
  FOREIGN KEY (clinicID) REFERENCES clinic(clinicID)
);

CREATE TABLE doctor_phoneNumber
(
  phoneNumber VARCHAR(50) NOT NULL,
  doctorID INT NOT NULL,
  PRIMARY KEY (phoneNumber, doctorID),
  FOREIGN KEY (doctorID) REFERENCES doctor(doctorID)
);

CREATE TABLE patient_phoneNumber
(
  phoneNumber VARCHAR(50) NOT NULL,
  patientID INT NOT NULL,
  PRIMARY KEY (phoneNumber, patientID),
  FOREIGN KEY (patientID) REFERENCES patient(patientID)
);

CREATE TABLE appointment
(
  appointmentID INT NOT NULL,
  date DATE NOT NULL,
  time TIME NOT NULL,
  status VARCHAR(50) NOT NULL,
  patientID INT NOT NULL,
  doctorID INT NOT NULL,
  roomID INT NOT NULL,
  PRIMARY KEY (appointmentID),
  FOREIGN KEY (patientID) REFERENCES patient(patientID),
  FOREIGN KEY (doctorID) REFERENCES doctor(doctorID),
  FOREIGN KEY (roomID) REFERENCES room(roomID)
);
```



a. Tabel Entitas Utama
   - **CREATE TABLE doctor**
     - **Tujuan:** Menyimpan data profil setiap dokter.
     - **Kolom:**
       - doctorID INT NOT NULL: ID unik untuk dokter (Angka, Wajib diisi).
       - firstName VARCHAR(100) NOT NULL: Nama depan (Teks, Wajib diisi).
       - lastName VARCHAR(100) NOT NULL: Nama belakang (Teks, Wajib diisi).
       - specialization VARCHAR(100) NOT NULL: Spesialisasi dokter (Teks, Wajib diisi).
     - **Key:**
       - PRIMARY KEY (doctorID): Menetapkan doctorID sebagai kunci unik. Tidak boleh ada dua dokter dengan ID yang sama.

   - **CREATE TABLE patient**
     - **Tujuan:** Menyimpan data profil setiap pasien.
     - **Kolom:**
       - patientID INT NOT NULL: ID unik untuk pasien.
       - firstName, lastName, emailAddress: Info dasar pasien (Teks, Wajib diisi).
     - **Key:**
       - PRIMARY KEY (patientID): Menjadikan patientID sebagai pengenal unik.

   - **CREATE TABLE clinic**
     - **Tujuan:** Menyimpan data setiap klinik yang terdaftar di sistem.
     - **Kolom:**
       - clinicID INT NOT NULL: ID unik untuk klinik.
       - clinicName, city, address: Info lokasi dan nama klinik (Teks, Wajib diisi).
     - **Key:**
       - PRIMARY KEY (clinicID): Menjadikan clinicID sebagai pengenal unik.

b. Tabel Entitas Terkait
   - **CREATE TABLE room**
     - **Tujuan:** Menyimpan data ruangan yang ada di dalam sebuah klinik.
     - **Kolom:**
       - roomID INT NOT NULL: ID unik untuk ruangan.
       - roomName VARCHAR(100) NOT NULL: Nama atau nomor ruangan (misal: "Poli Gigi 1").
       - capacity INT NOT NULL: Kapasitas ruangan.
       - clinicID INT NOT NULL: ID klinik tempat ruangan ini berada.
     - **Key:**
       - PRIMARY KEY (roomID): Menjadikan roomID sebagai pengenal unik.

    - FOREIGN KEY (clinicID) REFERENCES clinic(clinicID): Ini adalah **aturan penting**. Kolom clinicID di tabel ini *harus* merujuk ke clinicID yang sudah ada di tabel clinic. Ini menciptakan **relasi 1-to-Many** (1 Klinik *memiliki* Banyak Ruangan).

c. Tabel Relasional

- **CREATE TABLE doctor_clinic**
  - **Tujuan:** Ini adalah *junction table* (tabel persimpangan). Ini memecahkan **masalah Many-to-Many**.
  - **Logika:** Satu dokter bisa bekerja di banyak klinik, dan satu klinik bisa memiliki banyak dokter.
  - **Kolom:** Hanya berisi ID dari dua tabel yang dihubungkan.
  - **Key:**
    - PRIMARY KEY (doctorID, clinicID): Ini adalah *composite key*. Artinya, kombinasi doctorID dan clinicID harus unik. (Dokter A hanya bisa didaftarkan sekali di Klinik B).
    - FOREIGN KEY (doctorID) dan FOREIGN KEY (clinicID): Menghubungkan ke tabel doctor dan clinic.
- **CREATE TABLE doctor_phoneNumber**
  - **Tujuan:** Mengelola **atribut *multivalued***. Memungkinkan satu dokter punya banyak nomor telepon.
  - **Kolom:**
    - phoneNumber VARCHAR(50) NOT NULL: Menyimpan nomor telepon sebagai Teks (agar bisa menyimpan +, (,)).
    - doctorID INT NOT NULL: ID dokter pemilik nomor ini.
  - **Key:**
    - PRIMARY KEY (phoneNumber, doctorID): Kombinasi nomor dan ID dokter harus unik.
    - FOREIGN KEY (doctorID): Menghubungkan ke tabel doctor.
- **CREATE TABLE patient_phoneNumber**
  - **Tujuan:** Sama seperti doctor_phoneNumber, tapi untuk pasien.
  - **Logika:** Memungkinkan satu pasien mendaftarkan lebih dari satu nomor telepon.
  - **Key:** PRIMARY KEY (phoneNumber, patientID) dan FOREIGN KEY (patientID).

d. Tabel Transaksi Utama

- **CREATE TABLE appointment**
  - **Tujuan:** Ini adalah tabel **inti** dari sistem. Tabel ini mencatat setiap janji temu yang dibuat.
  - **Kolom:**
    - appointmentID INT NOT NULL: ID unik untuk janji temu.
    - date DATE NOT NULL: Tanggal janji temu.

- time TIME NOT NULL: Waktu janji temu.
- status VARCHAR(50) NOT NULL: Status (misal: "Dijadwalkan", "Selesai", "Batal").

- **Key:**
  - PRIMARY KEY (appointmentID): Menjadikan appointmentID sebagai pengenal unik.
  - FOREIGN KEY (patientID), (doctorID), (roomID): Setiap janji temu "mengikat" data dari tabel lain. Ini memberi tahu kita:
    - **Siapa** pasiennya (patientID).
    - **Siapa** dokternya (doctorID).
    - **Di mana** lokasinya (roomID).