**Nama : Andri Sarwono**

**NIM : 23EO50008**

**Prodi : Informatika**

**Entitas yang Diperlukan:**

1. **Nota**: Ini adalah entitas utama yang mewakili struk atau nota belanja.
   * Atribut: nota\_id (PK), tanggal, total\_harga, pelanggan\_id, metode\_pembayaran
2. **Pelanggan**: Entitas yang menyimpan data pelanggan (jika berlaku).
   * Atribut: pelanggan\_id (PK), nama\_pelanggan, alamat, telepon
3. **Item**: Ini adalah barang atau produk yang tercantum dalam nota.
   * Atribut: item\_id (PK), nama\_item, harga\_item
4. **Detail Nota**: Menyimpan informasi tentang item yang dibeli dalam setiap nota (relasi many-to-many antara Nota dan Item).
   * Atribut: detail\_nota\_id (PK), nota\_id (FK), item\_id (FK), jumlah, harga\_total
5. **Pembayaran**: Menyimpan informasi transaksi pembayaran.
   * Atribut: pembayaran\_id (PK), nota\_id (FK), jumlah\_dibayar, status\_pembayaran

**Proses Normalisasi:**

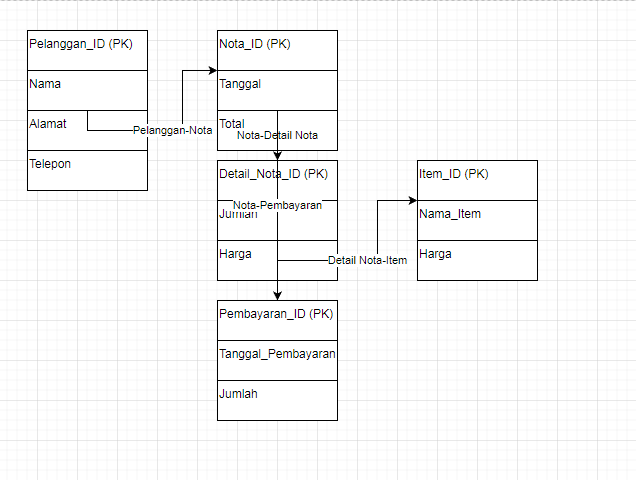
* **1NF (First Normal Form)**: Semua entitas harus memiliki atribut yang atomik, artinya data yang ada dalam sebuah kolom tidak bisa lebih dari satu nilai.
* **2NF (Second Normal Form)**: Setiap entitas harus bebas dari ketergantungan parsial (yaitu, semua atribut non-kunci harus bergantung pada seluruh kunci utama).
* **3NF (Third Normal Form)**: Tidak ada ketergantungan transitif antara atribut non-kunci dan kunci utama.

1. **ERD (Entity Relationship Diagram)** berikut adalah visualisasi hubungan antar entitas:
2. **Nota** berhubungan dengan **Pelanggan** (bisa many-to-one jika satu pelanggan bisa memiliki banyak nota, tetapi satu nota hanya dimiliki satu pelanggan).
3. **Nota** berhubungan dengan **Detail Nota** (one-to-many, satu nota memiliki banyak item dalam detailnya).
4. **Detail Nota** berhubungan dengan **Item** (many-to-one, satu item bisa muncul di banyak nota).
5. **Nota** berhubungan dengan **Pembayaran** (one-to-one atau one-to-many, satu nota bisa memiliki satu atau lebih transaksi pembayaran).

**Kardinalitas**:

* **Nota - Pelanggan**: Banyak nota dapat dimiliki oleh satu pelanggan, jadi hubungan ini adalah many-to-one.
* **Nota - Detail Nota**: Satu nota bisa memiliki banyak detail (produk yang dibeli), jadi one-to-many.
* **Detail Nota - Item**: Satu item dapat ada di banyak nota, jadi many-to-one.
* **Nota - Pembayaran**: Biasanya satu nota hanya memiliki satu pembayaran, jadi ini adalah hubungan one-to-one.

**DIAGRAM ERD :**

****

**Query :**

**Tabel-Tabel**:

-- Tabel Pelanggan

CREATE TABLE Pelanggan (

pelanggan\_id INT PRIMARY KEY,

nama\_pelanggan VARCHAR(100),

alamat VARCHAR(255),

telepon VARCHAR(15)

);

-- Tabel Nota

CREATE TABLE Nota (

nota\_id INT PRIMARY KEY,

tanggal DATE,

total\_harga DECIMAL(10,2),

pelanggan\_id INT,

metode\_pembayaran VARCHAR(50),

FOREIGN KEY (pelanggan\_id) REFERENCES Pelanggan(pelanggan\_id)

);

-- Tabel Item

CREATE TABLE Item (

item\_id INT PRIMARY KEY,

nama\_item VARCHAR(100),

harga\_item DECIMAL(10,2)

);

-- Tabel Detail Nota

CREATE TABLE DetailNota (

detail\_nota\_id INT PRIMARY KEY,

nota\_id INT,

item\_id INT,

jumlah INT,

harga\_total DECIMAL(10,2),

FOREIGN KEY (nota\_id) REFERENCES Nota(nota\_id),

FOREIGN KEY (item\_id) REFERENCES Item(item\_id)

);

-- Tabel Pembayaran

CREATE TABLE Pembayaran (

pembayaran\_id INT PRIMARY KEY,

nota\_id INT,

jumlah\_dibayar DECIMAL(10,2),

status\_pembayaran VARCHAR(50),

FOREIGN KEY (nota\_id) REFERENCES Nota(nota\_id)

);

**ACID Compliance :**

Untuk memastikan ACID compliance (Atomicity, Consistency, Isolation, Durability), kita akan menggunakan transaksi. Misalnya, untuk insert data transaksi pembayaran, kita perlu memastikan bahwa semua data yang berkaitan dengan nota dan pembayaran dimasukkan atau dihapus dalam satu unit yang konsisten.

START TRANSACTION;

-- Insert nota

INSERT INTO Nota (nota\_id, tanggal, total\_harga, pelanggan\_id, metode\_pembayaran)

VALUES (1, '2024-12-31', 150000, 101, 'Kartu Kredit');

-- Insert pembayaran

INSERT INTO Pembayaran (pembayaran\_id, nota\_id, jumlah\_dibayar, status\_pembayaran)

VALUES (1, 1, 150000, 'Lunas');

COMMIT;

**Manajemen Pengguna (User Management)**

Untuk membatasi akses pengguna sesuai dengan level akses yang diinginkan (super user, admin, user biasa), kita perlu membuat roles dan hak akses di database.

-- Membuat role untuk Super User

CREATE ROLE super\_user;

-- Membuat role untuk Admin

CREATE ROLE admin;

-- Membuat role untuk User Biasa

CREATE ROLE biasa\_user;

-- Memberikan hak akses

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO super\_user; -- Superuser dapat mengakses semua

GRANT INSERT, UPDATE ON Nota, Pembayaran TO admin; -- Admin hanya bisa insert dan update pada tabel tertentu

GRANT SELECT ON ALL TABLES IN SCHEMA public TO biasa\_user; -- User hanya bisa SELECT