Nama Anggota :

- Aloysius Pijar Hutama Indrianto (24/534591/PA/22675)
- Pison Golda Mountera (24/543770/PA/23107)
- Indratanaya Budiman (24/534784/PA/22683)

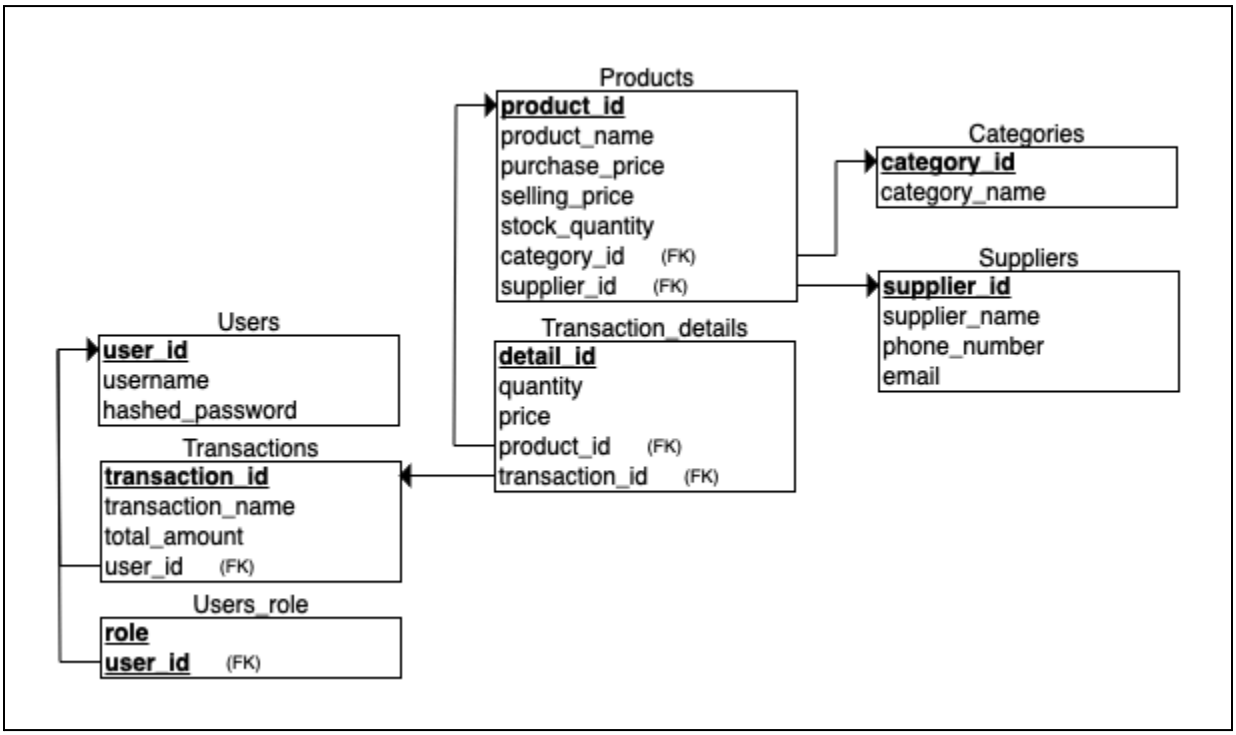
Kelas : KOMA

Logical Schema & SQL Implementation

Convert ERD into a relational schema, normalize up to 3NF, and implement with SQL (CREATE TABLE , keys, constraints).

1. **Relational Schema**

   Relational Schema dari database tersebut berupa berikut :



Dapat diperhatikan bahwa masing-masing entitas pada diagram memiliki hubungan saling merujuk dan menunjuk ke entitas lainnya melalui penggunaan foreign key. Hal ini terlihat jelas pada entitas Products dan Transaction_details, di mana Transaction_details merujuk ke Products dan Transactions untuk mencatat rincian setiap transaksi yang melibatkan produk tertentu. Hubungan ini menunjukkan adanya relasi *Many-to-One* antara Transaction_details dengan Products, serta antara Transaction_details dengan Transactions, karena satu produk atau transaksi dapat muncul pada banyak detail transaksi.

Sementara itu, pada entitas Users, atribut *role* bersifat *multivalued*, yang berarti seorang pengguna dapat memiliki lebih dari satu peran (misalnya sebagai admin dan kasir). Untuk

mengakomodasi hal tersebut, dibuat entitas terpisah bernama Users_role yang berfungsi sebagai tabel relasi antara user dan role. Entitas Users_role menunjuk ke Users melalui atribut user_id sebagai *foreign key*, sehingga setiap user dapat dikaitkan dengan satu atau lebih *role*(admin atau kasir) secara terstruktur dan fleksibel.

2. Normalization (3NF)

Sebelum melakukan normalisasi, perlu dipahami aturan normalisasi dari 1NF hingga 3NF.

1. 1NF: Setiap kolom berisi nilai atomik (tidak bisa dibagi lagi) dan setiap tabel memiliki Primary Key.

   Setiap tabel diberikan Primary Key yang jelas, yaitu pada tabel Roles, Users, Categories, Suppliers, Products, Transactions, Transaction_Details, User_Roles.

   Selain itu, bagian *multivalued attributes* seperti Role diatasi melalui tabel penghubung User_Roles yang menghubungkan Users dan Roles dalam hubungan *Many-to-Many*.

   Bagian *composite attribute* seperti contact_info pada Tabel Suppliers juga dibuat menjadi atomik dengan pembuatan kolom phone_number dan email.

2. 2NF: Tabel sudah 1NF dan semua atribut non-key harus bergantung penuh pada keseluruhan Primary Key. (Aturan ini utamanya relevan untuk tabel dengan *composite primary key* (PK gabungan))

   Dalam kasus sistem basis data kami, banyak tabel yang memang sudah memiliki 1 primary key (tunggal / single) sehingga otomatis tabel-tabel tersebut memenuhi aturan 2NF.

   Terdapat 1 tabel yang memiliki 2 Primary Key (*composite key*), yaitu tabel User_Roles. Akan tetapi, tabel tersebut tetap tidak melanggar 2NF karena tidak ada kolom lain yang memiliki ketergantungan secara parsial. Kedua kolom atribut yang ada pada tabel tersebut keduanya bersama menjadi Primary Key.

3. 3NF: Tabel sudah 2NF dan tidak boleh ada transitive dependency (dependensi transitif). Artinya, tidak boleh ada atribut non-key yang bergantung pada atribut non-key lainnya.

   Agar memenuhi 3NF, kita hapus kolom-kolom yang bersifat turunan, yaitu kolom yang dapat dikalkulasi dari kolom lainnya. Contoh kolom tersebut adalah subtotal dan total_amount. Keduanya dapat dikalkulasi dari kolom lainnya, yaitu sebagai berikut:

   - subtotal : quantity * price
   - total_amount : sum(subtotal)

   Selain itu, perlu dipastikan pula bahwa tidak ada ketergantungan antar kolom yang berstatus non-key. Secara sekilas, terkesan kolom price pada tabel Transaction_Details akan terlihat seperti bergantung pada product_id. Akan tetapi, hal tersebut tidaklah tepat karena kolom price pada tabel Transaction_Details berfungsi sebagai harga historis yang bergantung pada detail_id (saat transaksi tersebut terjadi).

3. SQL Implementation

Berdasarkan dari analisis yang telah dilakukan pada bagian Normalisasi, semua perhatian dan evaluasi tersebut diwujudkan dalam implementasi SQL menggunakan MySQL sebagai berikut:

```sql
CREATE DATABASE retail_manager;
USE retail_manager;
SHOW TABLES;

DROP TABLE IF EXISTS Transaction_Details;
DROP TABLE IF EXISTS Transactions;
DROP TABLE IF EXISTS Products;
DROP TABLE IF EXISTS Suppliers;
DROP TABLE IF EXISTS Categories;
DROP TABLE IF EXISTS User_Roles;
DROP TABLE IF EXISTS Users;
DROP TABLE IF EXISTS Roles;

CREATE TABLE Roles (
    role_id INT AUTO_INCREMENT PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL UNIQUE COMMENT 'example: admin, cashier'
);

CREATE TABLE Users (
    user_id BINARY(16) PRIMARY KEY,
    username VARCHAR(100) NOT NULL UNIQUE,
    hashed_password VARCHAR(255) NOT NULL COMMENT 'password must be hashed'
);

CREATE TABLE User_Roles (
    user_id BINARY(16) NOT NULL,
    role_id INT NOT NULL,
    PRIMARY KEY (user_id, role_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (role_id) REFERENCES Roles(role_id) ON DELETE RESTRICT
);

CREATE TABLE Categories (
    category_id BINARY(16) PRIMARY KEY,
    category_name VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE Suppliers (
    supplier_id BINARY(16) PRIMARY KEY,
    supplier_name VARCHAR(150) NOT NULL,
```

```
    phone_number VARCHAR(20),
    email VARCHAR(100)
);

CREATE TABLE Products (
    product_id BINARY(16) PRIMARY KEY,
    product_name VARCHAR(150) NOT NULL,
    purchase_price DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
    selling_price DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
    stock_quantity INT NOT NULL DEFAULT 0,
    category_id BINARY(16) NOT NULL,
    supplier_id BINARY(16) NOT NULL,
    FOREIGN KEY (category_id) REFERENCES Categories(category_id) ON DELETE
RESTRICT,
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id) ON DELETE
RESTRICT
);

CREATE TABLE Transactions (
    transaction_id BINARY(16) PRIMARY KEY,
    transaction_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    user_id BINARY(16) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE RESTRICT
);

CREATE TABLE Transaction_Details (
    detail_id BINARY(16) PRIMARY KEY,
    transaction_id BINARY(16) NOT NULL,
    product_id BINARY(16) NOT NULL,
    quantity INT NOT NULL CHECK (quantity > 0),
    price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (transaction_id) REFERENCES Transactions(transaction_id) ON
DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES Products(product_id) ON DELETE RESTRICT
);
```