Tentu, dengan senang hati. Merancang database untuk aplikasi persediaan barang yang "kompleks" berarti kita harus memikirkan lebih dari sekadar tabel barang dan stok. Sistem yang kompleks harus bisa menangani berbagai proses bisnis seperti pembelian, penjualan, perpindahan antar gudang, penyesuaian stok, varian produk, dan pelacakan riwayat secara detail.

Berikut adalah rancangan database yang komprehensif, dibagi menjadi beberapa modul utama untuk kemudahan pemahaman.

**Konsep Inti Desain**

1. **Pemisahan Produk Induk dan Varian:** Sebuah produk (misal: "Kaos Polos") bisa memiliki banyak varian (misal: Ukuran L Warna Merah, Ukuran M Warna Biru). Setiap varian memiliki SKU (Stock Keeping Unit), harga, dan stok yang unik.
2. **Stok sebagai Snapshot:** Tabel stok (stock\_levels) hanya menyimpan jumlah stok *saat ini* di lokasi tertentu.
3. **Pergerakan Stok sebagai Riwayat (Ledger):** Setiap perubahan stok (masuk, keluar, pindah) *wajib* dicatat di tabel stock\_movements. Tabel ini bersifat *append-only* (hanya bisa ditambah, tidak boleh diubah/dihapus) dan menjadi sumber kebenaran (source of truth) untuk audit.
4. **Modularitas:** Desain dipisahkan menjadi modul (Master Data, Pembelian, Penjualan, Gudang) agar mudah dikembangkan dan dipelihara.

**Diagram ERD (Entity-Relationship Diagram)**

Untuk visualisasi, berikut adalah diagram relasi antar tabel utama menggunakan sintaks Mermaid.

Cuplikan kode

erDiagram

PRODUCTS {

int product\_id PK

varchar name

text description

int category\_id FK

int brand\_id FK

}

PRODUCT\_VARIANTS {

int variant\_id PK

int product\_id FK

varchar sku UK

varchar variant\_name

decimal purchase\_price

decimal selling\_price

json attributes

}

WAREHOUSES {

int warehouse\_id PK

varchar name

text address

}

STOCK\_LEVELS {

int stock\_level\_id PK

int variant\_id FK

int warehouse\_id FK

int quantity\_on\_hand

int quantity\_committed

int quantity\_on\_order

}

STOCK\_MOVEMENTS {

bigint movement\_id PK

int variant\_id FK

int warehouse\_id FK

int quantity\_change

varchar movement\_type

int reference\_id

timestamp movement\_date

int user\_id FK

}

SUPPLIERS {

int supplier\_id PK

varchar name

varchar contact\_person

}

PURCHASE\_ORDERS {

int po\_id PK

int supplier\_id FK

date order\_date

varchar status

}

PURCHASE\_ORDER\_ITEMS {

int po\_item\_id PK

int po\_id FK

int variant\_id FK

int quantity

decimal unit\_price

}

CUSTOMERS {

int customer\_id PK

varchar name

varchar email

}

SALES\_ORDERS {

int so\_id PK

int customer\_id FK

date order\_date

varchar status

}

SALES\_ORDER\_ITEMS {

int so\_item\_id PK

int so\_id FK

int variant\_id FK

int quantity

decimal unit\_price

}

USERS {

int user\_id PK

varchar username

varchar password\_hash

int role\_id FK

}

CATEGORIES { int category\_id PK; varchar name; }

BRANDS { int brand\_id PK; varchar name; }

ROLES { int role\_id PK; varchar name; }

PRODUCTS ||--o{ PRODUCT\_VARIANTS : "has"

PRODUCT\_VARIANTS ||--o{ STOCK\_LEVELS : "has stock in"

WAREHOUSES ||--o{ STOCK\_LEVELS : "stores"

PRODUCT\_VARIANTS ||--o{ STOCK\_MOVEMENTS : "has history of"

WAREHOUSES ||--o{ STOCK\_MOVEMENTS : "occurs in"

USERS ||--o{ STOCK\_MOVEMENTS : "performed by"

SUPPLIERS ||--o{ PURCHASE\_ORDERS : "supplies"

PURCHASE\_ORDERS ||--o{ PURCHASE\_ORDER\_ITEMS : "contains"

PRODUCT\_VARIANTS }o--|| PURCHASE\_ORDER\_ITEMS : "is ordered"

CUSTOMERS ||--o{ SALES\_ORDERS : "buys"

SALES\_ORDERS ||--o{ SALES\_ORDER\_ITEMS : "contains"

PRODUCT\_VARIANTS }o--|| SALES\_ORDER\_ITEMS : "is sold"

CATEGORIES }o--|| PRODUCTS : "categorizes"

BRANDS }o--|| PRODUCTS : "from brand"

ROLES }o--|| USERS : "has role"

**Detail Struktur Tabel**

**Modul 1: Master Data (Data Inti)**

Tabel-tabel ini menyimpan informasi dasar yang jarang berubah.

1. **products (Produk Induk)**
   * product\_id (PK, INT, AUTO\_INCREMENT): ID unik untuk produk.
   * name (VARCHAR): Nama produk (Contoh: "Kemeja Lengan Panjang").
   * description (TEXT): Deskripsi lengkap produk.
   * category\_id (FK, INT): Merujuk ke tabel categories.
   * brand\_id (FK, INT): Merujuk ke tabel brands.
   * created\_at, updated\_at (TIMESTAMP).
2. **product\_variants (Varian Produk)**
   * variant\_id (PK, INT, AUTO\_INCREMENT): ID unik untuk setiap varian.
   * product\_id (FK, INT): Merujuk ke produk induk di products.
   * sku (VARCHAR, UNIQUE): Kode unik untuk varian ini (misal: "KMLP-BIRU-XL"). **Ini adalah kunci utama operasional.**
   * variant\_name (VARCHAR): Nama deskriptif varian (misal: "Biru, XL").
   * purchase\_price (DECIMAL): Harga beli dari supplier.
   * selling\_price (DECIMAL): Harga jual ke customer.
   * attributes (JSON): Untuk menyimpan atribut yang fleksibel (misal: {"warna": "Biru", "ukuran": "XL"}).
   * reorder\_level (INT): Batas minimum stok sebelum perlu restock.
   * barcode (VARCHAR, NULLABLE): Kode barcode jika ada.
3. **warehouses (Gudang/Lokasi Penyimpanan)**
   * warehouse\_id (PK, INT, AUTO\_INCREMENT): ID unik gudang.
   * name (VARCHAR): Nama gudang (misal: "Gudang Utama Jakarta", "Toko Cabang Bandung").
   * address (TEXT): Alamat lokasi.
   * is\_active (BOOLEAN): Status aktif/tidak.
4. **suppliers & customers (Pemasok & Pelanggan)**
   * supplier\_id / customer\_id (PK, INT): ID unik.
   * name, contact\_person, email, phone, address.
5. **Tabel Pendukung:** categories, brands, users, roles.

**Modul 2: Manajemen Stok (Inti Operasional)**

Ini adalah jantung dari sistem inventaris.

1. **stock\_levels (Tingkat Stok)**
   * stock\_level\_id (PK, INT, AUTO\_INCREMENT)
   * variant\_id (FK, INT): Varian produk yang dihitung.
   * warehouse\_id (FK, INT): Lokasi stok.
   * quantity\_on\_hand (INT): Jumlah fisik yang ada di gudang saat ini.
   * quantity\_committed (INT): Jumlah yang sudah dipesan oleh pelanggan tapi belum dikirim.
   * quantity\_on\_order (INT): Jumlah yang sudah dipesan dari supplier tapi belum diterima.
   * **Constraint:** UNIQUE(variant\_id, warehouse\_id) untuk memastikan satu baris per varian per gudang.
2. **stock\_movements (Riwayat Pergerakan Stok)** - **TABEL PALING PENTING**
   * movement\_id (PK, BIGINT, AUTO\_INCREMENT): ID unik untuk setiap transaksi.
   * variant\_id (FK, INT): Varian yang bergerak.
   * warehouse\_id (FK, INT): Gudang tempat pergerakan terjadi.
   * quantity\_change (INT): Jumlah perubahan. **Positif (+) untuk penambahan, Negatif (-) untuk pengurangan.**
   * movement\_type (ENUM/VARCHAR): Jenis pergerakan. Contoh:
     + GOODS\_RECEIPT: Penerimaan barang dari supplier.
     + SALES\_SHIPMENT: Pengiriman barang ke pelanggan.
     + STOCK\_ADJUSTMENT: Penyesuaian karena rusak/hilang/ditemukan.
     + TRANSFER\_OUT: Pindah barang keluar ke gudang lain.
     + TRANSFER\_IN: Pindah barang masuk dari gudang lain.
     + CUSTOMER\_RETURN: Pengembalian barang dari pelanggan.
   * reference\_id (INT): ID dari dokumen sumber (misal: po\_id, so\_id, adjustment\_id).
   * movement\_date (TIMESTAMP): Waktu pasti pergerakan terjadi.
   * user\_id (FK, INT): Siapa yang melakukan/mencatat transaksi.

**Modul 3: Proses Pembelian (Purchasing)**

1. **purchase\_orders (Pesanan Pembelian)**
   * po\_id (PK, INT, AUTO\_INCREMENT).
   * supplier\_id (FK, INT).
   * order\_date, expected\_delivery\_date (DATE).
   * status (ENUM: 'DRAFT', 'SUBMITTED', 'PARTIALLY\_RECEIVED', 'COMPLETED', 'CANCELLED').
   * total\_amount (DECIMAL).
2. **purchase\_order\_items (Detail Item Pesanan Pembelian)**
   * po\_item\_id (PK, INT, AUTO\_INCREMENT).
   * po\_id (FK, INT): Merujuk ke purchase\_orders.
   * variant\_id (FK, INT): Varian yang dipesan.
   * quantity\_ordered (INT).
   * unit\_price (DECIMAL): Harga beli saat itu.

*Catatan: Proses penerimaan barang (goods\_receipts) bisa dibuatkan tabel tersendiri untuk mencatat kapan dan berapa banyak barang yang diterima untuk sebuah PO, terutama jika pengiriman parsial diizinkan.*

**Modul 4: Proses Penjualan (Sales)**

Strukturnya sangat mirip dengan modul pembelian.

1. **sales\_orders (Pesanan Penjualan)**
   * so\_id (PK, INT, AUTO\_INCREMENT).
   * customer\_id (FK, INT).
   * order\_date (DATE).
   * status (ENUM: 'PENDING', 'PAID', 'SHIPPED', 'COMPLETED', 'CANCELLED').
   * shipping\_address (TEXT).
2. **sales\_order\_items (Detail Item Pesanan Penjualan)**
   * so\_item\_id (PK, INT, AUTO\_INCREMENT).
   * so\_id (FK, INT): Merujuk ke sales\_orders.
   * variant\_id (FK, INT): Varian yang dijual.
   * quantity (INT).
   * unit\_price (DECIMAL): Harga jual saat itu.

**Alur Kerja Logika Aplikasi**

* **Saat Barang Diterima (dari Supplier):**
  1. Buat entri di tabel goods\_receipts.
  2. Untuk setiap item yang diterima, buat entri di stock\_movements dengan movement\_type = 'GOODS\_RECEIPT' dan quantity\_change positif.
  3. Update quantity\_on\_hand di tabel stock\_levels (tambahkan).
  4. Update quantity\_on\_order di stock\_levels (kurangi).
* **Saat Barang Dikirim (ke Pelanggan):**
  1. Buat entri di tabel shipments.
  2. Buat entri di stock\_movements dengan movement\_type = 'SALES\_SHIPMENT' dan quantity\_change negatif.
  3. Update quantity\_on\_hand di stock\_levels (kurangi).
  4. Update quantity\_committed di stock\_levels (kurangi).
* **Untuk Menghitung Stok yang Tersedia (available\_stock):**
  1. Ini tidak disimpan di database, tetapi dihitung secara real-time:
  2. available\_stock = quantity\_on\_hand - quantity\_committed.

**Pertimbangan Tambahan untuk Kompleksitas**

* **Stock Opname/Penyesuaian:** Buat modul stock\_adjustments yang berisi header dan detail. Saat penyesuaian dilakukan, catat di tabel ini dan buat entri di stock\_movements dengan tipe STOCK\_ADJUSTMENT.
* **Perpindahan Antar Gudang:** Ini adalah transaksi dua langkah.
  1. Buat entri di stock\_movements untuk gudang asal (TRANSFER\_OUT, quantity\_change negatif).
  2. Buat entri di stock\_movements untuk gudang tujuan (TRANSFER\_IN, quantity\_change positif).
  3. Update stock\_levels di kedua gudang.
* **Serial/Batch Number:** Jika perlu melacak barang per unit atau per batch produksi, tambahkan tabel serial\_numbers atau batch\_numbers yang berelasi dengan product\_variants dan stock\_movements.
* **Indexing:** Pasang *index* pada semua kolom Foreign Key (FK) dan kolom yang sering dicari (seperti sku di product\_variants dan movement\_date di stock\_movements) untuk performa query yang cepat.

Rancangan ini menyediakan fondasi yang kuat, terstruktur, dan dapat diaudit untuk aplikasi manajemen persediaan barang yang kompleks dan skalabel.