

1. ERD


a. Tabel dan kolom

products	
product_id	INT
name	VARCHAR
description	VARCHAR
category	VARCHAR
price	INT
min_qty	INT
created_at	TIMESTAMP
updated_at	TIMESTAMP


product_id (PK)	: kode unik setiap produk
name	: nama produk
description	: deskripsi produk
category	: jenis produk
price	: harga produk
min_qty	: kuantitas minimal produk
created_at	: waktu saat produk diinput pada DB
updated_at	: waktu saat produk update pada DB

warehouse	
wh_id	INT
name	VARCHAR
location	VARCHAR
created_at	TIMESTAMP

wh_id (PK)	: kode unik setiap warehouse
name	: nama warehouse
location	: alamat warehouse
created_at	: waktu saat data diinput

stock	
stock 	INT
product_id	INT
wh_id	INT
qty	INT

stock (PK)	: kode unik stock
product_id (FK1)	: kode unik product
wh_id (FK2)	: kode unik warehouse
qty	: jumlah stock

supplier	
supplier_id 	INT
name	VARCHAR
contact	VARCHAR
phone	VARCHAR
email	VARCHAR
address	VARCHAR
created_at	VARCHAR

supplier_id (PK)	: kode unik supplier
name	: nama perusahaan supplier
contact	: nama PIC supplier
phone	: nomor supplier yang bisa dihubungi
email	: alamat email supplier yang bisa dihubungi
address	: alamat fisik supplier
created_at	: tanggal data diinput pada DB

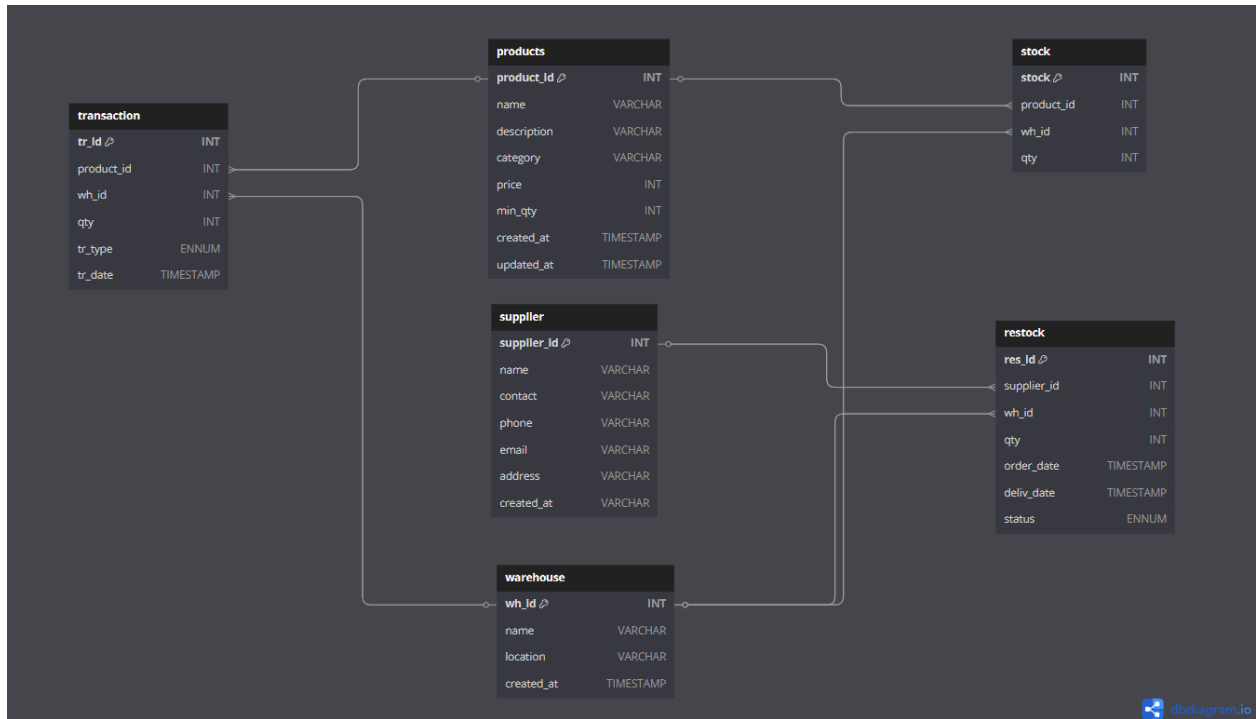
restock	
res_id 🔗	INT
supplier_id	INT
wh_id	INT
qty	INT
order_date	TIMESTAMP
deliv_date	TIMESTAMP
status	ENUM

res_id (PK)	: kode unik restock
supplier_id (FK1)	: kode unik supplier
wh_id (FK2)	: kode unik warehouse
qty	: jumlah barang yang dipesan
order_date	: tanggal pemesanan
deliv_date	: tanggal pengiriman
status	: status pengiriman (pending/terkirim/sudah sampai)

transaction	
tr_id 🔗	INT
product_id	INT
wh_id	INT
qty	INT
tr_type	ENUM
tr_date	TIMESTAMP

tr_id (PK)	: kode unik transaksi
product_id (FK1)	: kode unik produk
wh_id (FK2)	: kode unik warehouse
qty	: jumlah barang keluar
tr_type	: tipe transaksi (masuk/keluar)
tr_date	: tanggal transaksi

Diagram relasi masing-masing table



2. Query

- a. Menampilkan stok barang yang tersedia di semua gudang

```
SELECT

    p.product_id,

    p.name AS product_name,

    w.wh_id,

    w.name AS warehouse_name,

    s.qty AS stock_qty

FROM

    stock s

JOIN

    products p ON s.product_id = p.product_id

JOIN

    warehouse w ON s.wh_id = w.wh_id

ORDER BY

    p.product_id, w.wh_id;
```

- b. Menampilkan stok produk yang ada di bawah level minimum dan perlu di-restock

```
SELECT

    p.product_id,

    p.name AS product_name,

    w.name AS warehouse_name,

    s.qty AS current_stock,

    p.min_qty

FROM

    stock s
```

```
JOIN

    products p ON s.product_id = p.product_id

JOIN

    warehouse w ON s.wh_id = w.wh_id

WHERE

    s.qty < p.min_qty

ORDER BY

    current_stock ASC;
```

c. Menghitung total nilai inventaris di setiap gudang berdasarkan harga produk

```
SELECT

    w.wh_id,

    w.name AS warehouse_name,

    SUM(s.qty * p.price) AS total_inventory_value

FROM

    stock s

JOIN

    products p ON s.product_id = p.product_id

JOIN

    warehouse w ON s.wh_id = w.wh_id

GROUP BY

    w.wh_id, w.name

ORDER BY

    total_inventory_value DESC;
```

d. Menghasilkan laporan transaksi masuk dan keluar per bulan untuk satu tahun terakhir

```

SELECT

    TO_CHAR(tr.tr_date, 'YYYY-MM') AS month,

    SUM(CASE WHEN tr.qty > 0 THEN tr.qty ELSE 0 END) AS total_in,

    SUM(CASE WHEN tr.qty < 0 THEN ABS(tr.qty) ELSE 0 END) AS total_out

FROM

    transaction tr

WHERE

    tr.tr_date >= NOW() - INTERVAL '1 year'

GROUP BY

    TO_CHAR(tr.tr_date, 'YYYY-MM')

ORDER BY

    month;

```

Optimalisasi yang dapat dilakukan adalah menggunakan *indexing* pada beberapa *primary key* dan kolom dan tabel yang sering menggunakan **JOIN**, **WHERE**, dan **ORDER BY**, pada *case* ini kolom dan tabel yang kemungkinan sering diakses adalah jumlah **stok produk(s.qty)**, **tanggal transaksi(tr.tr_date)**, dan **jumlah minimal dari produk(p.min_qty)**, hal ini dikarenakan saat rekap, pembuatan laporan, dan audit akan sering mengambil nilai dari kolom dan tabel tersebut.

Kemudian melakukan automasi, seperti saat restock yang nantinya menghasilkan rekomendasi produk mana saja yang perlu restock secara berkala, salah metodenya menggunakan cronjob jika OS yang digunakan adalah linux based. Berikut contoh cronjob yang digunakan untuk mengeksekusi query setiap jam 9 pagi pada hari ke-10 setiap bulannya.

```
0 9 */10 */1 * psql -U [USERNAME DB] -d [NAMA DB] -f [PATH]/recommend_restock.sql
```

Dan terakhir menggunakan *Materialized View* jika ingin melihat laporan bulanan diluar waktu pelaporan rutin, tujuannya ialah untuk tidak membebani karena pada *Materialized View* hanya menampilkan dan tidak menyimpan hasil dari query.

```

CREATE MATERIALIZED VIEW monthly_transaction_summary AS

SELECT

```

```

    TO_CHAR(tr.tr_date, 'YYYY-MM') AS month,

    SUM(CASE WHEN tr.qty > 0 THEN tr.qty ELSE 0 END) AS total_in,

    SUM(CASE WHEN tr.qty < 0 THEN ABS(tr.qty) ELSE 0 END) AS total_out

FROM

    transaction tr

WHERE

    tr.tr_date >= NOW() - INTERVAL '1 year'

GROUP BY

    TO_CHAR(tr.tr_date, 'YYYY-MM');

```

3. Performance Tuning

Seperti yang sudah dijelaskan sebelumnya pada nomor 2, untuk meningkatkan performa dapat menggunakan *Indexing* pada kolom dan tabel yang sering menggunakan **JOIN**, **WHERE**, dan **ORDER BY**, berikut beberapa indexing query dan penjelasannya

```

// pada tabel stock

CREATE INDEX idx_stock_product_wh ON stock(product_id, wh_id); // digunakan untuk mempermudah
pencarian stok per barang per Gudang

// pada tabel transaction

CREATE INDEX idx_products_min_qty ON products(min_qty);

CREATE INDEX idx_products_category ON products(category); //mempermudah pencarian produk yang perlu
restock dan melakukan filter per category

// pada tabel restock

CREATE INDEX idx_restock_status_date ON restock(status, order_date); //untuk filter order yang masih
pending atau terlalu kirim

```

Untuk melakukan pemantauan dapat langsung dari query SQL seperti `pg_stat_activity` pada PostgreSQL dan `SHOW ENGINE INNODB STATUS;` pada MySQL atau menggunakan tools monitoring seperti PgHero.

Untuk penanganan deadlock bisa melakukan pengaturan Lock Timeout dengan query `SET innodb_lock_wait_timeout = 5;` atau bisa retry logic seperti menggunakan script Node.js atau Python pada sisi client secara otomatis apabila dalam rentang waktu yang sudah ditentukan client tidak bisa mengakses database.

4. Partitioning dan Maintenance

Partisi dilakukan pada tabel yang akan terus bertambah setiap tahunnya dan dalam jumlah besar seperti tabel **transaction** dan **restock**

```
// partisi pada tabel transaction

CREATE TABLE transaction_2025_01 PARTITION OF transaction

    FOR VALUES FROM ('2025-01-01') TO ('2025-02-01'); //membuat partisi pada bulan januari

// partisi pada tabel restock

CREATE TABLE restock_2025 PARTITION OF restock

    FOR VALUES FROM ('2025-01-01') TO ('2026-01-01'); //melakukan partisi selama satu tahun
```

Maintenance DB dapat dilakukan dengan reindexing secara rutin dengan **REINDEX DATABASE [NAMA DB]**; dan automasi secara rutin dengan menggunakan cronjob

```
0 6 * * */7 psql -U [USERNAME DB] -d [NAMA DB] -f [PATH]/reindex.sql // melakukan reindexing setiap
7 hari pada jam 6 pagi
```

Menggunakan APM tools seperti Grafana dan Prometheus, dan melakukan backup otomatis seperti melakukan Snapshot.