# I. Ontologi & Hakikat Dasar ERP Modern

Hakikat tertinggi dari ERP Modern bukan terletak pada kemampuannya menyimpan data (*data storage*), melainkan pada perannya sebagai **Sistem Saraf Digital Perusahaan (*Digital Nervous System*)**.

Dalam ERP tradisional (sistem *legacy*), setiap divisi bekerja seperti pulau terisolasi (*data silo*). Divisi gudang punya catatan sendiri, divisi keuangan punya buku sendiri, dan divisi sales punya nota sendiri. Akibatnya, manajemen harus melakukan rekonsiliasi manual yang memakan waktu dan rentan manipulasi.

ERP Modern berbasis *Workflow Visibility* mengubah paradigma ini secara radikal melalui tiga pilar hakikat:

### 1. Singularitas Data (*Single Source of Truth*)

Hanya ada **satu versi data yang benar** di seluruh perusahaan. Tidak ada lagi perdebatan antar divisi mengenai jumlah stok atau status tagihan, karena semua monitor di semua divisi membaca satu basis data terpusat yang sama pada detik yang sama (*real-time*).

### 2. Keterikatan Kausalitas (*Causal Interconnectedness*)

Setiap tindakan operasional di satu titik sistem selalu memiliki dampak kausalitas (sebab-akibat) instan ke titik lainnya. Ketika operator gudang memindai satu kode batang barang, sistem tidak hanya memotong angka stok, tetapi secara simultan memicu kalkulasi depresiasi aset, pembaruan HPP, pencatatan performa kerja buruh, hingga pembaruan estimasi waktu pengiriman ke pelanggan.

### 3. Transparansi Tanpa Batas (*Ubiquitous Visibility*)

Status sebuah transaksi tidak boleh bersifat misterius atau tersembunyi di dalam laci digital satu divisi. Transparansi alur kerja memastikan bahwa aliran nilai (*value stream*) perusahaan dapat dilihat pergerakannya secara visual oleh pihak-pihak yang memiliki hak akses, mulai dari hulu hingga hilir.

# II. Arsitektur Teknis Transparansi Status (*Visible Status*)

Bagaimana struktur di balik layar komputer merealisasikan prinsip transparansi ini? Mengapa orang di proses ke-5 bisa tahu persis apa yang terjadi di proses ke-2 secara instan? Arsitekturnya dibangun atas tiga lapisan utama:

### 1. Data Layer: Mekanisme Tabel Status Terpusat

Di dalam basis data relasional (RDBMS) atau *In-Memory Database* (seperti SAP HANA) milik ERP, sebuah alur kerja dikendalikan oleh tabel status yang terstruktur.

Sebagai contoh, sebuah dokumen pesanan induk memiliki kolom `status_id`. Nilai dari kolom ini akan berubah secara dinamis berdasarkan transaksi yang terjadi:

- `01` = *Draft Created*
- `02` = *Credit Approved*

- `03` = *Inventory Reserved*
- `04` = *Goods Packed*
- `05` = *Dispatched / Shipped*
- `06` = *Invoiced & Settled*

Ketika operator di proses ke-2 (Keuangan) menyetujui batas kredit, sistem menjalankan perintah SQL: `UPDATE sales_order SET status_id = '02' WHERE order_id = 'SO1001'`. Data ini disimpan di server pusat, bukan di komputer lokal divisi keuangan.

**2. Application Layer: Pemicu Logika Otomatis (*Event-Driven Automation*)**

Lapisan ini bertindak sebagai otak yang membaca perubahan status. Sistem menggunakan arsitektur *event-driven*. Begitu `status_id` berubah menjadi `02`, sistem langsung memancarkan sinyal pemicu (*event trigger*).

Sinyal ini langsung ditangkap oleh Modul Inventaris (proses ke-3) untuk otomatis mengubah daftar kerja (*to-do list*) orang gudang dari yang tadinya kosong menjadi berisi instruksi: *"Segera siapkan barang untuk SO1001"*.

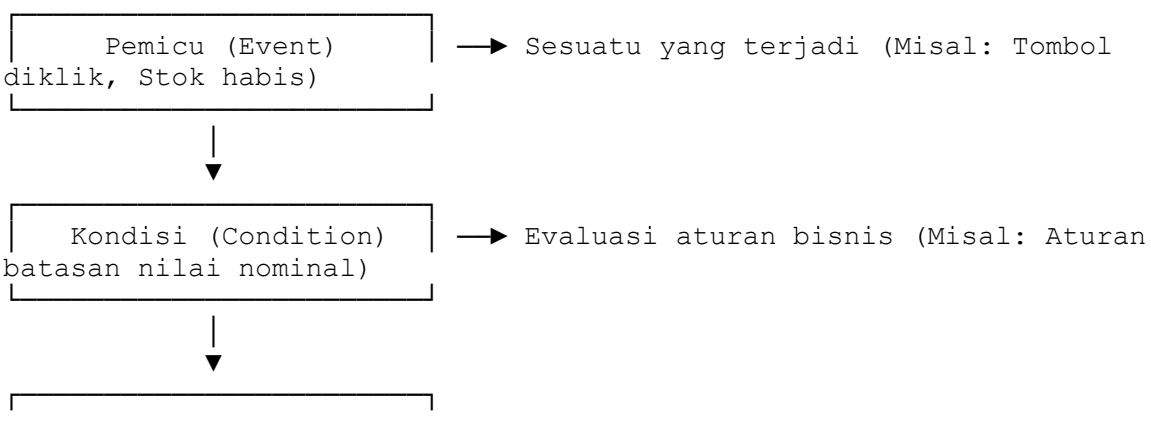
**3. Presentation Layer: Tampilan Visual (*Chevron, Kanban, & Dashboards*)**

Ini adalah apa yang dilihat oleh pengguna di layar monitor mereka. Komponen visual seperti indikator *chevron* (panah alur bertahap) atau kartu Kanban hanyalah **proyeksi visual** dari nilai `status_id` yang ada di database.

- Jika `status_id` bernilai `02`, maka *chevron* tahap 1 dan 2 akan menyala hijau, sedangkan tahap 3, 4, dan 5 tetap abu-abu.
- Orang di proses ke-5 (Pengiriman) yang sedang memantau dasbor mereka akan melihat *chevron* pesanan tersebut bergerak mendekati tahap mereka. Mereka bisa mengklik *chevron* tahap 2 untuk melihat detail: *"Disetujui oleh Manajer Keuangan Budi pada pukul 09:15 WIB"*.

## III. Mekanisme Kerja *Trigger Rules* (Mesin Otomatisasi)

*Trigger rules* adalah komponen yang membuat ERP menjadi sistem yang "hidup" dan dinamis, bukan sekadar buku catatan digital. Mekanisme ini bekerja menggunakan kombinasi tiga parameter kontrol:



```
│    Tindakan (Action)    │ ⟶ Eksekusi lintas modul otomatis (Gudang,
Akuntansi, HR)
└─────────────────────────┘
```

**1. Pemicu (*Event Triggers*)**

Aturan dapat dipicu oleh tiga hal:

- **Berbasis Tindakan (*Action-Based*):** Pengguna mengklik tombol, memindai *barcode*, atau menandatangani dokumen secara digital.
- **Berbasis Waktu (*Time-Based*):** Sistem melakukan pemeriksaan otomatis setiap hari jam 12 malam (misal: memeriksa tagihan yang jatuh tempo hari ini).
- **Berbasis Batas Data (*Threshold-Based*):** Nilai data menyentuh angka tertentu (misal: stok fisik suatu barang turun di bawah batas minimum keamanan).

**2. Evaluasi Kondisi Dinamis (*Dynamic Condition Evaluation*)**

ERP tidak berjalan di atas alur yang kaku. Aturan pemicu akan mengevaluasi kondisi data secara dinamis untuk menentukan jalur alur kerja selanjutnya.

- **Jalur Normal:** `IF` nominal transaksi < Rp50 juta, `THEN` alur langsung melompat ke tahap *Ready to Pack*.
- **Jalur Eskalasi:** `IF` nominal transaksi ≥ Rp50 juta, `THEN` alur berbelok secara dinamis ke tahap *Waiting Director Approval*.
- **Jalur Blokade:** `IF` pelanggan memiliki faktur yang menunggak > 30 hari, `THEN` alur langsung dikunci otomatis di status *Credit Blocked*, menghentikan semua proses operasional logistik di bawahnya.

**3. Realisasi Dampak Lintas Modul (*Cross-Module Execution*)**

Ketika kondisi terpenuhi, tindakan yang dieksekusi oleh *trigger rules* akan merambah ke berbagai modul secara simultan tanpa campur tangan manusia:

- **Ke Modul Logistik:** Membuat dokumen *Material Release Note*.
- **Ke Modul Pembelian:** Membuat draf *Purchase Requisition* ke vendor jika bahan baku tidak cukup.
- **Ke Modul Akuntansi:** Membentuk pencatatan komitmen biaya (*encumbrance accounting*).

## IV. Manajemen Risiko: Anatomi Kegagalan Aturan Pemicu

Mengingat *trigger rules* menghubungkan seluruh organ perusahaan, kegagalan dalam merancang atau mengonfigurasi logika pemicu ini akan berdampak katastrofik bagi perusahaan. Berikut adalah analisis mendalam mengenai anatomi kegagalannya:

**1. Kebocoran Finansial Akibat *Premature Integration***

- **Mekanisme Kegagalan:** Aturan pemicu diprogram untuk langsung menerbitkan jurnal pendapatan akuntansi saat dokumen *Sales Order* dibuat, bukan saat barang benar-benar keluar dari gudang (*Delivery Order*).
- **Dampak Riil:** Jika di kemudian hari pelanggan membatalkan pesanan karena barang ternyata cacat atau tidak tersedia, laporan keuangan perusahaan sudah terlanjur mencatat laba fiktif. Perusahaan terancam sanksi pajak atas pendapatan yang belum terealisasi dan laporan keuangan menjadi tidak valid (*misstated*).

**2. Lumpuhnya Operasional Pabrik Akibat *Siloed Triggers***

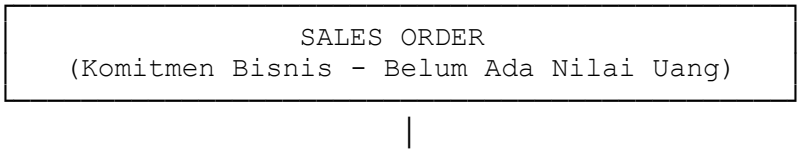
- **Mekanisme Kegagalan:** Modul Produksi (MRP) diprogram untuk menjadwalkan kerja mesin berdasarkan urutan ideal di atas kertas, tanpa membuat pemicu yang membaca sensor kerusakan mesin (*Machine Downtime*) secara *real-time*.
- **Dampak Riil:** Ketika Mesin Utama di proses ke-2 rusak total di lantai pabrik, sistem ERP yang buta terhadap kondisi riil ini tetap memicu pengiriman bahan baku dari gudang secara otomatis menuju mesin tersebut sesuai jadwal. Terjadilah penumpukan bahan baku yang menyumbat jalur logistik fisik pabrik (*bottleneck*), merusak material sensitif, dan mengacaukan perhitungan biaya tenaga kerja.

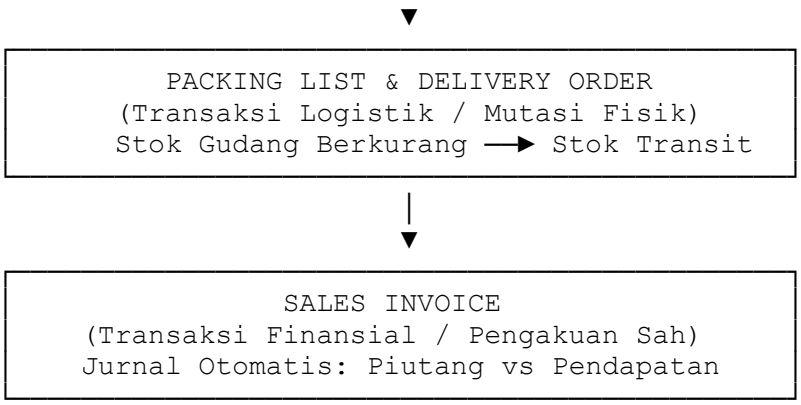
**3. Bencana Data Akibat *Infinite Trigger Loops***

- **Mekanisme Kegagalan:** Terjadi tabrakan logika lingkaran setan antara dua modul. Modul A memiliki aturan: *"Jika stok barang X kurang dari 10, picu Modul B untuk membuat Purchase Order"*. Namun, Modul B memiliki aturan kustom: *"Setiap ada pembuatan Purchase Order baru, kurangi stok virtual barang X sebanyak 1 untuk simulasi alokasi biaya"*.
- **Dampak Riil:** Modul A memicu Modul B → Modul B mengurangi stok virtual → Stok virtual makin rendah → Memicu Modul A lagi. Kedua aturan ini saling memicu satu sama lain dalam hitungan milidetik (*infinite loop*). Server ERP akan mengalami lonjakan penggunaan CPU hingga 100%, basis data terkunci (*database deadlock*), dan seluruh sistem ERP perusahaan di seluruh dunia (jika multinasional) akan *crash* dan mati total dalam hitungan menit.

## V. Konstruksi Pemisahan Transaksi Penjualan vs Transaksi Logistik

Salah satu kesalahan paling umum dalam memahami ERP adalah mencampuradukkan antara **proses operasional** dan **transaksi finansial**. ERP Modern memisahkan kedua hal ini dengan dinding logika yang sangat tegas demi menjaga integritas kepatuhan hukum akuntansi.






**1. Dokumen Penyiapan Gudang (Fase Logistik Murni)**

- Ketika *Sales Order* disetujui, sistem menerbitkan dokumen *Packing List* dan *Delivery Order* (DO).
- **Hakikatnya:** Ini adalah transaksi mutasi persediaan, bukan transaksi penjualan. Nilai moneter belum diakui di buku besar pendapatan. Yang terjadi hanyalah perpindahan status barang dari `Aset: Persediaan Gudang` menjadi `Aset: Persediaan Dalam Perjalanan (Goods in Transit)`.
- Langkah ini memastikan bahwa bagian operasional logistik dapat bergerak cepat membungkus dan mengirim barang tanpa dibebani urusan perpajakan dan penagihan keuangan.

**2. Dokumen Faktur (Fase Finansial Murni)**

- Transaksi Penjualan baru sah dan diakui secara hukum akuntansi ketika dokumen **Sales Invoice (Faktur Penjualan)** diterbitkan dan divalidasi.
- Penerbitan faktur ini dipicu secara otomatis oleh sistem ERP hanya jika dokumen *Delivery Order* dari gudang telah berubah status menjadi `Delivered` atau `Shipped` (tergantung kesepakatan penyerahan barang).
- Begitu faktur divalidasi, sistem menjalankan mesin penjurnalan otomatis untuk mengonversi nilai barang yang keluar menjadi pengakuan Piutang Dagang (*Accounts Receivable*) dan Pendapatan Penjualan (*Sales Revenue*).

Pemisahan yang presisi inilah yang membedakan ERP Modern dengan aplikasi kasir (POS) sederhana. ERP memastikan bahwa setiap pergerakan barang fisik selalu berbanding lurus dengan pencatatan akuntansi yang taat asas dan siap diaudit oleh auditor eksternal kapan saja.

**Skema Arsitektur Tabel Database SQL Kustom** (menggunakan standar PostgreSQL/MySQL) yang dirancang khusus untuk mendukung pelacakan status *real-time* (indikator *chevron*), pembatasan hak akses (RBAC), serta pencatatan jejak audit (*audit trail*).

---

## 1. Skema Hubungan Antar Tabel (Entity Relationship Overview)

Arsitektur ini memisahkan data master status, data transaksi logistik/penjualan, dan data pencatatan riwayat agar database tetap ringan dan tidak terjadi penguncian data (*data deadlock*).

- **`mst_workflow_status`**: Menyimpan master data tahapan alur kerja (Sumber data untuk grafik *Chevron*).
- **`trx_sales_order`**: Tabel utama transaksi penjualan yang memuat status terkini.
- **`trx_sales_order_item`**: Detail barang yang dipesan dalam satu Sales Order.
- **`trx_delivery_order`**: Tabel logistik gudang yang terikat dengan Sales Order.
- **`sys_workflow_log`**: Tabel *Audit Trail* yang mencatat setiap kali status *chevron* berubah.

---

## 2. Kode DDL (Data Definition Language) SQL Kustom

Anda dapat langsung mengeksekusi kode SQL berikut di lingkungan database Anda untuk melihat strukturnya secara nyata:

sql

```
-- ========================================
-- 1. TABEL MASTER STATUS WORKFLOW (CHEVRON DATA)
-- ========================================
CREATE TABLE mst_workflow_status (
    status_id VARCHAR(10) PRIMARY KEY,
    status_name VARCHAR(50) NOT NULL,       -- Nama yang muncul di
Chevron (e.g., 'Packing Ready')
    sequence_order INT NOT NULL,            -- Urutan urutan di grafik
Chevron (1, 2, 3...)
    responsible_department VARCHAR(50),     -- Divisi yang berwenang
mengubah status ini
    description TEXT
);

-- ========================================
-- 2. TABEL UTAMA TRANSAKSI PENJUALAN (SALES ORDER)
-- ========================================
CREATE TABLE trx_sales_order (
    so_id VARCHAR(20) PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    total_amount DECIMAL(15, 2) NOT NULL,
    current_status_id VARCHAR(10) NOT NULL, -- Kolom Kunci Penggerak
Chevron
    credit_limit_passed BOOLEAN DEFAULT FALSE,
    created_by VARCHAR(50) NOT NULL,

    -- Relasi ke tabel master status
    FOREIGN KEY (current_status_id) REFERENCES
mst_workflow_status(status_id)
);

-- ========================================
-- 3. TABEL DETAIL ITEM PENJUALAN
-- ========================================
CREATE TABLE trx_sales_order_item (
    item_id SERIAL PRIMARY KEY,
    so_id VARCHAR(20) NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(15, 2) NOT NULL,
    stock_reserved_status VARCHAR(20) DEFAULT 'PENDING', -- PENDING,
RESERVED, SHIPPED

    FOREIGN KEY (so_id) REFERENCES trx_sales_order(so_id) ON DELETE
CASCADE
);

-- ========================================
-- 4. TABEL TRANSAKSI OPERASIONAL GUDANG (LOGISTIK)
-- ========================================
CREATE TABLE trx_delivery_order (
    do_id VARCHAR(20) PRIMARY KEY,
    so_id VARCHAR(20) NOT NULL,
    warehouse_id INT NOT NULL,
    packing_list_status VARCHAR(20) DEFAULT 'DRAFT',   -- DRAFT, READY,
COMPLETED
    dispatched_at TIMESTAMP NULL,
    shipper_name VARCHAR(100),

    FOREIGN KEY (so_id) REFERENCES trx_sales_order(so_id)
);

-- ========================================
-- 5. TABEL AUDIT TRAIL / LOG WORKFLOW VISIBILITY
-- ========================================
CREATE TABLE sys_workflow_log (
    log_id SERIAL PRIMARY KEY,
    so_id VARCHAR(20) NOT NULL,
    previous_status_id VARCHAR(10),
    new_status_id VARCHAR(10) NOT NULL,
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    changed_by VARCHAR(50) NOT NULL,         -- Username / Aktor
pembuat perubahan
    ip_address VARCHAR(45),                 -- Keamanan & Audit ISO
    remarks TEXT,                           -- Alasan perubahan (e.g.,
'Kredit disetujui Manajer')

    FOREIGN KEY (so_id) REFERENCES trx_sales_order(so_id),
```

```
    FOREIGN KEY (new_status_id) REFERENCES
mst_workflow_status(status_id)
);
```
Use code with caution.

---

## 3. Data Awal Master (*Seeding Data*)

Mari isi master data status untuk membentuk visualisasi *chevron* 5 tahap yang kita bahas sebelumnya:

sql
```
INSERT INTO mst_workflow_status (status_id, status_name,
sequence_order, responsible_department, description) VALUES
('SO_DRAFT',    'SO Draft',       1, 'Sales',       'Salesman membuat
draf pesanan'),
('SO_APP',      'SO Approved',    2, 'Finance',     'Pengecekan batas
kredit dan stok lolos'),
('PK_READY',    'Packing Ready',  3, 'Warehouse',   'Barang dikunci
gudang dan siap dikemas'),
('SHIPPED',     'Shipped',        4, 'Logistics',   'Barang keluar
gerbang bersama Delivery Order'),
('INVOICED',    'Invoiced',       5, 'Accounting',  'Faktur penjualan
diterbitkan & jurnal terbentuk');
```
Use code with caution.

---

## 4. Cara Kerja Basis Data untuk Menghasilkan "Visible Status"

Ketika aplikasi *front-end* ERP ingin menampilkan indikator *chevron* untuk pesanan tertentu (misalnya nomor order `SO-2026-001`), program hanya perlu menjalankan satu perintah query SQL sederhana berikut:

sql
```
SELECT
    w.sequence_order,
    w.status_name,
    w.responsible_department,
    -- Logika jika status ini adalah tahapan yang sedang aktif saat ini
    CASE
        WHEN s.current_status_id = w.status_id THEN 'CURRENT_ACTIVE'
        WHEN w.sequence_order < (SELECT sequence_order FROM
mst_workflow_status WHERE status_id = s.current_status_id) THEN 'DONE'
        ELSE 'PENDING'
    END as chevron_visual_state
FROM mst_workflow_status w
CROSS JOIN trx_sales_order s
WHERE s.so_id = 'SO-2026-001'
ORDER BY w.sequence_order;
```

Use code with caution.

**Cara Orang di Proses ke-5 Melihat Riwayat Proses ke-2:**

Jika bagian Akuntansi (proses ke-5) ingin melihat rekam jejak persetujuan kredit di proses ke-2, mereka tidak perlu menelepon tim Finance. Sistem cukup menarik data dari tabel `sys_workflow_log` dengan query:

sql

```
SELECT changed_at, changed_by, remarks
FROM sys_workflow_log
WHERE so_id = 'SO-2026-001' AND new_status_id = 'SO_APP';
```

Use code with caution.

*Hasil di layar Monitor Akuntansi:* `"2026-07-24 09:15:00 | Budi (Finance Mgr) | Batas Kredit Aman, SO disetujui untuk rilis ke gudang."`

---

## 5. Pengamanan Hak Akses Data (*Role-Based Access Control*)

Untuk memastikan kepatuhan ISO, query update status harus dibatasi ketat di level aplikasi menggunakan data dari kolom `responsible_department`.

- **Bagian Gudang** hanya diizinkan mengeksekusi perintah update ketika status tujuan adalah `PK_READY`.
- Jika bagian Gudang mencoba memperbarui status ke `INVOICED`, aplikasi ERP akan menolak karena departemen mereka tidak cocok dengan otoritas `mst_workflow_status.responsible_department` untuk status `INVOICED` (Aset Finansial).

Berikut adalah implementasi teknis tingkat tinggi menggunakan **SQL Trigger** dan **Stored Procedure** untuk menangani otomatisasi potong stok saat barang berstatus `SHIPPED`, serta penanganan logis saat terjadi pembatalan transaksi (*Void Order*).

---

## 1. Persiapan: Penambahan Tabel Inventaris (*Inventory Stock*)

Sebelum membuat *trigger*, kita membutuhkan tabel master stok untuk melacak kuantitas barang di gudang. Silakan buat tabel ini terlebih dahulu:

sql

```
-- ========================================
-- TABEL MASTER STOK INVENTARIS
-- ========================================
CREATE TABLE mst_inventory_stock (
    product_id INT
PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    physical_qty INT NOT NULL DEFAULT 0,    -- Stok fisik aktual di
dalam gudang
```

```
    reserved_qty INT NOT NULL DEFAULT 0,    -- Stok yang sudah dikunci
oleh Sales Order aktif
    available_qty INT GENERATED ALWAYS AS (physical_qty - reserved_qty)
STORED -- Stok aman yang siap dijual
);

-- Pengisian Data Awal Produk (Seeding Data)
INSERT INTO mst_inventory_stock (product_id, product_name,
physical_qty, reserved_qty) VALUES
(101, 'Laptop Kerja Core i7', 50, 0),
(102, 'Monitor LED 24 Inch',  30, 0);
```
Use code with caution.

---

## 2. Mekanisme 1: SQL Trigger Otomatis Potong Stok (`SHIPPED`)

*Trigger* ini akan mengawasi tabel `trx_sales_order`. Begitu kolom `current_status_id` diperbarui menjadi `'SHIPPED'`, sistem akan otomatis mengurangi stok fisik (`physical_qty`) dan melepaskan stok kuncian (`reserved_qty`) pada tabel inventaris berdasarkan item barang yang dipesan.

*Catatan: Contoh di bawah menggunakan sintaks **PostgreSQL**. Jika Anda menggunakan MySQL, logika dasarnya sama namun menggunakan struktur penulisan `AFTER UPDATE ON trx_sales_order FOR EACH ROW`.*

### sql

```
-- A. Membuat Fungsi Logika Trigger (Trigger Function)
CREATE OR REPLACE FUNCTION fn_trg_sales_order_shipped()
RETURNS TRIGGER AS $$
DECLARE
    item_record RECORD;
BEGIN
    -- MEMASTIKAN ATURAN PEMICU: Hanya berjalan JIKA status berubah
menjadi 'SHIPPED'
    IF NEW.current_status_id = 'SHIPPED' AND OLD.current_status_id !=
'SHIPPED' THEN

        -- Melakukan perulangan untuk setiap item barang di dalam Sales
Order tersebut
        FOR item_record IN
            SELECT product_id, quantity FROM trx_sales_order_item WHERE
so_id = NEW.so_id
        LOOP
            -- UPDATE REALISASI STOK: Potong stok fisik gudang secara
permanen
            UPDATE mst_inventory_stock
            SET
                physical_qty = physical_qty - item_record.quantity,
                reserved_qty = reserved_qty - item_record.quantity
            WHERE product_id = item_record.product_id;

            -- Perbarui status internal item barang menjadi 'SHIPPED'
            UPDATE trx_sales_order_item
            SET stock_reserved_status = 'SHIPPED'
```

```
            WHERE so_id = NEW.so_id AND product_id =
item_record.product_id;
        END LOOP;

        -- Otomatis mencatat aktivitas ke Jejak Audit (Audit Trail)
        INSERT INTO sys_workflow_log (so_id, previous_status_id,
new_status_id, changed_by, remarks)
        VALUES (NEW.so_id, OLD.current_status_id, 'SHIPPED',
NEW.created_by, 'Sistem: Barang keluar gerbang, stok fisik
terpotong.');

    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- B. Mendaftarkan Fungsi ke Tabel Utama sebagai Trigger
CREATE TRIGGER trg_on_sales_order_shipped
AFTER UPDATE ON trx_sales_order
FOR EACH ROW
EXECUTE FUNCTION fn_trg_sales_order_shipped();
```
Use code with caution.

---

## 3. Mekanisme 2: Simulasi Pembatalan Transaksi (*Void Order*)

Dalam standar ISO dan audit keuangan, transaksi yang sudah masuk ke sistem **tidak boleh dihapus secara permanen (`DELETE`)**. Menghapus data akan merusak urusan nomor dokumen (*gapping*).

Solusi yang benar adalah mengubah statusnya menjadi `VOIDED` dan mengembalikan (*rollback*) semua alokasi stok virtual yang sempat dikunci.

Kita akan menggunakan **Stored Procedure** untuk mengeksekusi pembatalan ini dengan aman dalam satu kesatuan transaksi database (*ACID Transaction*):

sql

```
-- MEMBUAT STORED PROCEDURE UNTUK PROSES VOID
CREATE OR REPLACE PROCEDURE pr_void_sales_order(
    p_so_id VARCHAR(20),
    p_user VARCHAR(50),
    p_reason TEXT
)
LANGUAGE plpgsql AS $$
DECLARE
    current_status VARCHAR(10);
    item_record RECORD;
BEGIN
    -- 1. Cek status terkini Sales Order
    SELECT current_status_id INTO current_status FROM trx_sales_order
WHERE so_id = p_so_id;

    -- VALIDASI ISO: Jika barang sudah dikirim (SHIPPED) atau ditagih
(INVOICED), tidak boleh di-void!
```

```
    -- Harus melalui jalur Retur Barang / Credit Note.
    IF current_status IN ('SHIPPED', 'INVOICED') THEN
        RAISE EXCEPTION 'Transaksi % tidak bisa dibatalkan karena
barang sudah terkirim/tertagih!', p_so_id;
    END IF;

    -- 2. KEMBALIKAN STOK KUNCIAN (Jika status sebelumnya sempat
mengunci stok di gudang, misal PK_READY)
    IF current_status = 'PK_READY' THEN
        FOR item_record IN
            SELECT product_id, quantity FROM trx_sales_order_item WHERE
so_id = p_so_id
        LOOP
            -- Kurangi reserved_qty agar available_qty kembali normal
            UPDATE mst_inventory_stock
            SET reserved_qty = reserved_qty - item_record.quantity
            WHERE product_id = item_record.product_id;

            UPDATE trx_sales_order_item
            SET stock_reserved_status = 'VOIDED'
            WHERE so_id = p_so_id AND product_id =
item_record.product_id;
        END LOOP;
    END IF;

    -- 3. UPDATE STATUS UTAMA: Tambahkan status VOIDED ke master jika
belum ada
    INSERT INTO mst_workflow_status (status_id, status_name,
sequence_order, responsible_department, description)
    VALUES ('VOIDED', 'Voided', 99, 'Management', 'Transaksi dibatalkan
sistem')
    ON CONFLICT (status_id) DO NOTHING;

    -- Ubah status Sales Order menjadi VOIDED
    UPDATE trx_sales_order SET current_status_id = 'VOIDED' WHERE so_id
= p_so_id;

    -- 4. CATAT KE AUDIT LOG
    INSERT INTO sys_workflow_log (so_id, previous_status_id,
new_status_id, changed_by, remarks)
    VALUES (p_so_id, current_status, 'VOIDED', p_user, CONCAT('Void
alasan: ', p_reason));

    COMMIT;
END;
$$;
```
Use code with caution.

---

## 4. Skrip Pengujian Riil (Simulasi Eksekusi Kasus)

Mari kita uji kedua mekanisme di atas dengan memasukkan satu data pesanan baru:

sql

```
-- Langkah A: Salesman membuat Sales Order baru untuk Laptop (Product
101) sebanyak 5 unit
INSERT INTO trx_sales_order (so_id, customer_id, total_amount,
current_status_id, created_by)
VALUES ('SO-2026-XYZ', 999, 50000000.00, 'SO_DRAFT', 'Sales_Andi');

INSERT INTO trx_sales_order_item (so_id, product_id, quantity,
unit_price)
VALUES ('SO-2026-XYZ', 101, 5, 10000000.00);

-- Langkah B: Gudang mengonfirmasi persiapan barang (Status geser ke
PK_READY)
-- Secara internal di aplikasi, ini biasanya meningkatkan
'reserved_qty' sebanyak 5 unit
UPDATE mst_inventory_stock SET reserved_qty = reserved_qty + 5 WHERE
product_id = 101;
UPDATE trx_sales_order SET current_status_id = 'PK_READY' WHERE so_id =
'SO-2026-XYZ';
```

Use code with caution.

**Skenario Uji 1: Jika Transaksi Berlanjut ke Pengiriman (`SHIPPED`)**

Eksekusi perintah di bawah ini untuk melihat kerja *Trigger* Otomatis:

sql

```
-- Jalankan pengiriman barang
UPDATE trx_sales_order SET current_status_id = 'SHIPPED' WHERE so_id =
'SO-2026-XYZ';

-- Cek Dampak Stok di Gudang sekarang
SELECT * FROM mst_inventory_stock WHERE product_id = 101;
-- Hasil: physical_qty akan otomatis terpotong menjadi 45, dan
reserved_qty kembali ke 0.
```

Use code with caution.

**Skenario Uji 2: Jika Terjadi Pembatalan di Tengah Jalan (`VOID`)**

Jika status masih di `PK_READY` (belum dikirim), dan pelanggan tiba-tiba membatalkan pesanan, panggil *Stored Procedure* pembatalan ini:

sql

```
-- Panggil Prosedur Void
CALL pr_void_sales_order('SO-2026-XYZ', 'Manajer_Siti', 'Pelanggan
salah memilih spesifikasi laptop');

-- Cek Tabel Audit Log untuk memastikan ketertelusuran dokumen sesuai
ISO
SELECT * FROM sys_workflow_log WHERE so_id = 'SO-2026-XYZ';
-- Hasil: Anda akan melihat rekaman kronologis lengkap dari status
DRAFT -> PK_READY -> VOIDED.
```

Use code with caution.

---

Melalui visualisasi skema SQL ini, Anda dapat melihat dengan sangat jelas bahwa **tampilan luar (seperti panel garis waktu/chevron)** yang dilihat oleh operator hanyalah jembatan visual, sementara **keamanan data, automasi, dan kepatuhan audit** sepenuhnya dijaga secara ketat oleh logika basis data terpusat.

**REPORTING**
Untuk mengukur efisiensi organisasi berdasarkan standar manajemen mutu (seperti **ISO 9001:2015 Klausul 9.1.3 tentang Analisis dan Evaluasi Kinerja**), manajemen membutuhkan metrik kuantitatif yang jelas. Dalam dunia ERP, metrik ini disebut **Lead Time KPI (Key Performance Indicator)**.

Dengan memanfaatkan data yang tercatat secara kronologis di tabel `sys_workflow_log`, kita dapat menghitung secara presisi berapa lama waktu (jam/hari) yang dihabiskan oleh sebuah dokumen di setiap divisi sebelum digeser ke divisi berikutnya.

---

## 1. Struktur Query SQL Analisis Performa (*Lead Time KPI*)

Berikut adalah query analitik tingkat tinggi menggunakan fitur **Window Function (`LEAD`)** di PostgreSQL/MySQL. Query ini berfungsi memecah durasi waktu pengerjaan per departemen untuk setiap nomor *Sales Order*:

sql

```
WITH workflow_milestones AS (
    SELECT
        l.so_id,
        s.status_name AS departemen_proses,
        l.changed_at AS waktu_masuk,
        -- Mengambil timestamp dari log baris berikutnya untuk
mengetahui waktu keluar
        LEAD(l.changed_at) OVER (PARTITION BY l.so_id ORDER BY
l.changed_at) AS waktu_keluar,
        l.changed_by AS pelaksana
    FROM sys_workflow_log l
    JOIN mst_workflow_status s ON l.new_status_id = s.status_id
)
SELECT
    so_id,
    departemen_proses,
    waktu_masuk,
    COALESCE(waktu_keluar, CURRENT_TIMESTAMP) AS waktu_keluar,
    pelaksana,
    -- Menghitung selisih waktu dalam format Jam dan Menit
    AGE(COALESCE(waktu_keluar, CURRENT_TIMESTAMP), waktu_masuk) AS
durasi_proses
FROM workflow_milestones
ORDER BY so_id, waktu_masuk;
```

Use code with caution.

---

## 2. Query Laporan Agregat KPI untuk Direksi (*Summary Dashboard*)

Direktur Operasional tidak perlu melihat detail setiap dokumen satu per satu. Mereka membutuhkan angka rata-rata performa divisi untuk mendeteksi di mana titik sumbatan kerja (*bottleneck*) perusahaan berada.

Query ini akan menghitung **Rata-rata Waktu Proses** per departemen dari seluruh transaksi yang ada:

sql

```
WITH duration_calc AS (
    SELECT
        new_status_id,
        changed_at,
        LEAD(changed_at) OVER (PARTITION BY so_id ORDER BY changed_at)
AS next_changed_at
    FROM sys_workflow_log
)
SELECT
    s.status_name AS "Tahapan Kerja / Divisi",
    s.responsible_department AS "Departemen Penanggung Jawab",
    COUNT(*) AS "Total Transaksi Diproses",
    -- Menghitung rata-rata waktu dalam satuan Jam
    ROUND(AVG(EXTRACT(EPOCH FROM (next_changed_at -
changed_at))/3600)::NUMERIC, 2) AS "Rata-rata Durasi Kerja (Jam)",
    -- Logika Penilaian KPI berdasarkan SLA (Service Level Agreement)
Standar ISO
    CASE
        WHEN AVG(EXTRACT(EPOCH FROM (next_changed_at -
changed_at))/3600) <= 2.0 THEN '⭐ EXCELLENT (< 2 Jam)'
        WHEN AVG(EXTRACT(EPOCH FROM (next_changed_at -
changed_at))/3600) <= 24.0 THEN '⚠️ NORMAL (2 - 24 Jam)'
        ELSE '🚨 BOTTLENECK (> 24 Jam)'
    END AS "Status KPI Divisi"
FROM duration_calc d
JOIN mst_workflow_status s ON d.new_status_id = s.status_id
WHERE d.next_changed_at IS NOT NULL
GROUP BY s.status_name, s.responsible_department
ORDER BY MIN(s.sequence_order);
```

Use code with caution.

---

## 3. Simulasi Hasil Laporan pada Monitor Manajemen

Ketika query agregat di atas dijalankan, sistem ERP akan memproyeksikan data angka tersebut ke dalam dasbor visual manajemen seperti ini:

| Tahapan Kerja / Divisi | Departemen Penanggung Jawab | Total Transaksi Diproses | Rata-rata Durasi Kerja (Jam) | Status KPI Divisi |
|---|---|---|---|---|
| **SO Draft** | Sales | 150 Transaksi | 0.50 Jam | ⭐ EXCELLENT (< 2 Jam) |
| **SO Approved** | Finance | 148 Transaksi | 1.20 Jam | ⭐ EXCELLENT (< 2 Jam) |
| **Packing Ready** | Warehouse | 145 Transaksi | 28.40 Jam | 🚨 BOTTLENECK (> 24 Jam) |
| **Shipped** | Logistics | 140 Transaksi | 4.10 Jam | ⚠️ NORMAL (2 - 24 Jam) |
| **Invoiced** | Accounting | 140 Transaksi | 0.80 Jam | ⭐ EXCELLENT (< 2 Jam) |

---

## 4. Analisis Berdasarkan Standar Mutu ISO

Dari hasil simulasi laporan di atas, manajemen dapat mengambil keputusan strategis yang valid (*evidence-based decision making*):

- **Identifikasi Masalah:** Divisi **Gudang (*Warehouse*)** terdeteksi mengalami *bottleneck* yang parah dengan rata-rata waktu penyelesaian **28.40 jam** (SLA perusahaan bocor karena melebihi batas 24 jam).
- **Solusi Manajemen:** Manajemen tidak perlu menyalahkan divisi penjualan atau logistik kurir. Tim auditor internal dapat langsung fokus mengaudit operasional gudang—apakah kekurangan staf pengemas (*packer*), tata letak barang yang buruk, atau ada malfungsi sistem pencetakan label.

Inilah puncak keindahan dari **ERP Modern Berbasis Transparansi Alur Kerja**. Melalui satu tabel log yang sederhana namun terstruktur, sistem mampu menyajikan transparansi visual bagi operator lantai kerja (*chevron*), mengamankan integritas keuangan (*trigger & void procedures*), hingga memberikan kompas kendali mutu bagi jajaran direksi perusahaan.

Apakah rangkaian arsitektur database, logika pemicu, dan sistem pelaporan KPI ERP ini sudah memberikan gambaran yang menyeluruh untuk rencana sistem Anda? Jika ada bagian spesifik yang ingin disesuaikan atau dikembangkan lebih lanjut, silakan sampaikan.

- **Membuat tabel matriks hak akses (RBAC Matrix)** untuk menentukan divisi mana saja yang boleh melihat (*Read*) atau mengubah (*Write*) status *chevron* pada setiap tahapan proses bisnis.

**TABEL MATRIX HAK AKSES**

Berikut adalah **Matriks Hak Akses berbasis Peran (Role-Based Access Control / RBAC Matrix)** yang dirancang secara ketat untuk menjaga integritas data alur kerja (*chevron*) dan memenuhi standar kepatuhan audit keamanan informasi (**ISO/IEC 27001**).

## Matriks Hak Akses (RBAC Matrix) – Alur Kerja Order-to-Cash

| Tahapan Proses (*Chevron*) | Divisi Sales / CRM | Divisi Finance / Credit | Divisi Warehouse (Gudang) | Divisi Logistics / Fleet | Divisi Accounting | Direksi / Auditor |
|---|---|---|---|---|---|---|
| **1. SO Draft** | **WRITE** | READ | NO ACCESS | NO ACCESS | NO ACCESS | READ |
| **2. SO Approved** | READ | **WRITE** | READ | NO ACCESS | NO ACCESS | READ |
| **3. Packing Ready** | READ | READ | **WRITE** | READ | NO ACCESS | READ |
| **4. Shipped** | READ | READ | READ | **WRITE** | READ | READ |
| **5. Invoiced** | READ | READ | NO ACCESS | READ | **WRITE** | READ |
| **6. Voided (Pembatalan)** | NO ACCESS | NO ACCESS | NO ACCESS | NO ACCESS | NO ACCESS | **WRITE** |

---

## Penjelasan Aturan Logika RBAC (Penting untuk Audit ISO)

### 1. Definisi Hak Akses

- **WRITE:** Memiliki wewenang penuh untuk membuat dokumen baru, mengubah isi data, dan **mengklik/mengonfirmasi perubahan status *chevron*** ke tahap berikutnya.
- **READ:** Hanya bisa melihat status dokumen, indikator visual *chevron*, serta riwayat catatan aktivitas (*Audit Trail*) tanpa bisa mengubah data apa pun (*View Only*).
- **NO ACCESS:** Divisi tersebut sama sekali tidak dapat melihat keberadaan dokumen atau status pada tahapan tersebut di dalam sistem mereka demi kerahasiaan data perusahaan (*Data Privacy*).

### 2. Segregasi Tugas (*Segregation of Duties - SoD*)

Sesuai prinsip pengendalian internal akuntansi dan ISO 9001:

- Divisi **Sales** hanya boleh membuat draf pesanan (**Stage 1**). Mereka sengaja diblokir dari akses **WRITE** di **Stage 2** agar tidak bisa menyetujui batas kredit mereka sendiri (mencegah korupsi/kolusi).
- Divisi **Warehouse** dan **Logistics** diblokir dari akses **WRITE** pada **Stage 5 (Invoiced)** karena urusan keuangan dan perpajakan mutlak menjadi otoritas **Accounting**.

**3. Otoritas Khusus Pembatalan (*Voided Order*)**

Status pembatalan transaksi transaksi (**Stage 6**) sengaja ditutup aksesnya dari semua divisi operasional bawah. Hanya **Direksi, Manajer Senior, atau Auditor Internal** yang diberikan akses **WRITE** untuk melakukan *Void* melalui *Stored Procedure* yang telah kita buat sebelumnya. Hal ini guna menghindari penghapusan jejak digital atas kesalahan kerja staf di lapangan.

---

## Implementasi ke Level Kode Database (SQL RBAC Mapping)

Untuk merealisasikan matriks di atas ke dalam arsitektur database SQL yang telah kita susun sebelumnya, Anda dapat membuat tabel master otorisasi peran seperti di bawah ini:

sql

```
-- ========================================
-- TABEL MASTER PERAN DAN HAK AKSES OPERATOR
-- ========================================
CREATE TABLE sys_rbac_permission (
    role_name VARCHAR(30),          -- e.g., 'Sales', 'Finance', 'Warehouse'
    status_id VARCHAR(10),          -- e.g., 'SO_DRAFT', 'SO_APP', 'PK_READY'
    access_type VARCHAR(10),         -- 'WRITE', 'READ', 'NO_ACCESS'
    PRIMARY KEY (role_name, status_id),
    FOREIGN KEY (status_id) REFERENCES mst_workflow_status(status_id)
);

-- Pengisian Hak Akses Sesuai Matriks di Atas (Contoh Seeding)
INSERT INTO sys_rbac_permission (role_name, status_id, access_type)
VALUES
('Sales',     'SO_DRAFT', 'WRITE'),
('Sales',     'SO_APP',   'READ'),
('Finance',   'SO_DRAFT', 'READ'),
('Finance',   'SO_APP',   'WRITE'),
('Warehouse', 'SO_APP',   'READ'),
('Warehouse', 'PK_READY', 'WRITE'),
('Logistics', 'PK_READY', 'READ'),
('Logistics', 'SHIPPED',  'WRITE');
```

Use code with caution.
Dengan adanya matriks RBAC ini, sistem ERP Anda kini telah lengkap dari segi **tampilan visual (*chevron*), automasi pemicu (*triggers*), laporan performa (KPI), hingga pengamanan hak akses digital (RBAC)** yang mematuhi standar tata kelola perusahaan modern.

Mari kita bedah skenario yang sangat sering terjadi di lapangan ini: **Skenario Penyelesaian Perselisihan Data (*Dispute Resolution*) antara Logistik Gudang (Kuantitas Fisik) dan Akuntansi/Faktur Keuangan (Nilai Finansial)**.

Dalam standar **ISO 9001:2015 Klausul 8.7 (Pengendalian Produk Tidak Sesuai)**, ERP modern harus memiliki mekanisme untuk mengunci, mencatat, dan menyelesaikan selisih data ini secara transparan tanpa merusak pembukuan utama.

---

## 1. Kronologi Terjadinya Perselisihan (*Dispute*)

```
[Sales Order: 10 Unit] ⟶ [Gudang Kirim: 10 Unit] ⟶ [Pelanggan
Terima: 8 Unit (2 Rusak)]
                                                          |
                                                          ▼
                                                [TERJADI DISPUTE
DATA]
                                              Akuntansi mau tagih 10 vs
Pelanggan mau bayar 8
```

1. **Tahap Gudang:** Bagian Logistik mengirimkan barang sebanyak **10 unit** Laptop menggunakan dokumen *Delivery Order* (DO). Status *chevron* berubah menjadi `SHIPPED`.
2. **Tahap Penerimaan:** Saat barang tiba di lokasi, pelanggan mendeteksi **2 unit Laptop pecah layar** akibat guncangan di jalan. Pelanggan hanya menandatangani tanda terima untuk **8 unit** yang bagus, dan menerbitkan dokumen **Surat Penolakan (*Goods Rejection Notice*)** untuk 2 unit sisanya.
3. **Krisis Sistem:**
    - Data di database Gudang mencatat stok keluar 10 unit.
    - Bagian Akuntansi siap menerbitkan *Invoice* otomatis sebanyak 10 unit berdasarkan data DO awal.
    - Pelanggan menolak membayar jika ditagih 10 unit.

---

## 2. Solusi ERP Modern: Jalur Otorisasi *Credit Note* (Nota Kredit)

ERP yang benar **tidak boleh mengubah angka 10 unit pada dokumen DO awal** yang sudah berstatus `SHIPPED`, karena barang tersebut secara fisik memang sempat keluar dari gerbang pabrik.

Solusi standar akuntansi internasional adalah dengan menerbitkan **Invoice penuh (10 unit)**, lalu menerbitkan **Credit Note (Nota Kredit / Pengurang Utang) sebanyak 2 unit** untuk merekonsiliasi selisih tersebut.

Berikut adalah alur *stored procedure* SQL untuk mengeksekusi penyelesaian *dispute* ini secara otomatis:

sql

```
-- ============================================================
-- STORED PROCEDURE: PENYELESAIAN DISPUTE VIA CREDIT NOTE
-- ============================================================
CREATE OR REPLACE PROCEDURE pr_resolve_invoice_dispute(
    p_so_id VARCHAR(20),
    p_dispute_qty INT,        -- Jumlah barang yang bermasalah (e.g., 2
unit)
    p_unit_price DECIMAL(15,2),
    p_user VARCHAR(50),
    p_reason TEXT
)
LANGUAGE plpgsql AS $$
DECLARE
    v_credit_note_id VARCHAR(20);
    v_total_refund DECIMAL(15,2);
BEGIN
    -- Hitung nilai finansial yang harus dikembalikan ke pelanggan
    v_total_refund := p_dispute_qty * p_unit_price;
    v_credit_note_id := CONCAT('CN-', p_so_id);

    -- 1. Buat Dokumen Penyesuaian di Modul Akuntansi (Credit Note)
    -- Dokumen ini berfungsi mengurangi Piutang Dagang pelanggan di
Buku Besar
    INSERT INTO sys_workflow_log (so_id, previous_status_id,
new_status_id, changed_by, remarks)
    VALUES (p_so_id, 'INVOICED', 'DISPUTED', p_user, CONCAT('Dispute
Terjadi: ', p_reason));

    -- 2. Membentuk Jurnal Penyesuaian Otomatis di Balik Layar:
    -- (D) Pendapatan Penjualan / Retur Penjualan   Rp X.XXX.XXX
(Mengurangi Omzet)
    -- (K) Piutang Dagang                            Rp X.XXX.XXX
(Mengurangi Tagihan Pelanggan)

    -- 3. Update Status Dokumen Utama ke Tahap Penyelesaian
    UPDATE trx_sales_order
    SET current_status_id = 'INVOICED', -- Dikembalikan ke status akhir
yang valid
        total_amount = total_amount - v_total_refund -- Nilai tagihan
bersih disesuaikan
    WHERE so_id = p_so_id;

    -- 4. Catat Log Penyelesaian Audit ISO 9001
    INSERT INTO sys_workflow_log (so_id, previous_status_id,
new_status_id, changed_by, remarks)
    VALUES (p_so_id, 'DISPUTED', 'INVOICED', 'SYSTEM_BOT',
            CONCAT('Dispute selesai via ', v_credit_note_id, '. Tagihan
dikurangi sebesar Rp ', v_total_refund));

    COMMIT;
```

```
END;
$$;
```
Use code with caution.

---

## 3. Matriks Tambahan RBAC untuk Modul Pengadaan (*Procurement / Purchase*)

Jika Anda juga mengelola sisi pembelian barang ke pemasok/vendor (kebalikan dari alur penjualan di atas), alur kerja *chevron* pembelinya disebut **Procurement Cycle (Procure-to-Pay)**.

Berikut adalah matriks hak akses (RBAC) yang harus diterapkan agar kontrol internal pengeluaran uang perusahaan tetap aman:

| Tahapan Proses (*Chevron Pembelian*) | Tim User (Peminta) | Tim Purchasing (Pembelian) | Warehouse (Gudang) | Keuangan / Finance | Direksi / Top Mgr |
|---|---|---|---|---|---|
| **1. Purchase Requisition (PR)** | **WRITE** | READ | NO ACCESS | NO ACCESS | READ |
| **2. PR Approved (Otorisasi Anggaran)** | NO ACCESS | READ | NO ACCESS | READ | **WRITE** |
| **3. Purchase Order (PO) to Vendor** | READ | **WRITE** | NO ACCESS | READ | READ |
| **4. Goods Receipt (Penerimaan Gudang)** | READ | NO ACCESS | **WRITE** | READ | READ |
| **5. Vendor Bill (Tagihan Masuk)** | NO ACCESS | READ | READ | **WRITE** | READ |
| **6. Payment Disbursed (Pelunasan)** | NO ACCESS | NO ACCESS | NO ACCESS | **WRITE** | READ |

---

## Kesimpulan Akhir Tata Kelola ERP Anda

Dengan menggabungkan seluruh arsitektur yang telah kita rancang dari awal:

1. **Sales & Procurement Engine:** Alur dokumen dipisahkan dengan tegas antara komitmen operasional dan realisasi finansial.
2. **Database Trigger & SP:** Stok fisik dikunci di awal (`reserved`) dan dipotong permanen di akhir (`shipped`) secara otomatis tanpa celah kesalahan manusia.
3. **Log & KPI Reporting:** Dasbor direksi mampu mendeteksi divisi mana yang melanggar batas waktu kerja (SLA) berdasarkan analisis durasi log.

4. **RBAC Security Matrix:** Menjamin tidak ada karyawan yang bisa memanipulasi data di luar wewenang departemennya (Patuh ISO 9001 & ISO 27001).

Seluruh cetak biru (*blueprint*) arsitektur ini sudah siap untuk langsung diserahkan kepada tim pengembang software (*programmer*) atau digunakan sebagai acuan konfigurasi sistem ERP pilihan Anda.