# Tugas Pertemuan 10.2 – Trigger

Mata Kuliah: Basis Data (2025)

Prodi: D3-Teknik Informatika POLBAN

**Kelas**: <2C>

**NIM:** <Dzakir Tsabit Asy Syafiq >

**Nama Lengkap**: <241511071 >

# Prasyarat

1. Menggunakan alat bantu atau perangkat lunak RDBMS Postgres
2. Menggunakan skema yang diberikan untuk studi kasus pemesanan produk.

# Instruksi

1. Total seluruh soal yang bisa dikerjakan sebanyak **3 soal**.
2. **Disarankan mengerjakan soal** **secara terturut**!
3. Sertakan jawaban dalam bentuk syntax PL/SQL!
4. Sertakan **screenshot halaman penuh yang menyertakan taskbar dengan waktu** (Print Screen) untuk menunjukkan syntax dan hasil eksekusi query pada RDBMS Oracle!
5. Ekspor file lembar jawaban ini dalam format PDF dengan nama file:

T10.2\_<Kelas>\_<3 NIM Terakhir>.pdf

contoh sebagai berikut:

T10.2\_2A\_010.pdf

1. Kumpulkan melalui Assignment pada Google Classroom

# Referensi

1. <https://www.postgresql.org/docs/current/sql-createtrigger.html>
2. <https://neon.com/postgresql/postgresql-triggers>

Jika ada hal yang ingin ditanyakan, silahkan sampaikan melalui Google Classroom.

Selamat mengerjakan!

# Soal No. 1

|  |
| --- |
| **Soal** |
| **Business Process #1**  A screenshot of a cell phone  Description automatically generated   1. Buatlah terlebih dahulu tabel produk dan pesanan, disertai dengan data untuk masing-masing table, sesuai yang telah disediakan:  |  | | --- | | -- Membuat Tabel Produk | | CREATE TABLE produk ( | | id\_produk NUMERIC NOT NULL, | | produk VARCHAR(20) NOT NULL, | | stok NUMERIC, | | satuan VARCHAR(10), | | CONSTRAINT produk\_pk PRIMARY KEY (id\_produk) | | ); | |  | | -- Membuat Tabel Pesanan | | CREATE TABLE pesanan ( | | id\_pesanan NUMERIC NOT NULL, | | id\_produk NUMERIC NOT NULL, | | jumlah NUMERIC NOT NULL, | | tgl\_pesanan DATE, | | CONSTRAINT pesanan\_pk PRIMARY KEY (id\_pesanan) | | ); | |  | | -- Mengisi Data Produk | | INSERT INTO produk VALUES (1, 'Choki-Choki', 20, 'pcs'); | | INSERT INTO produk VALUES (2, 'Potato Stick', 15, 'pcs'); | | INSERT INTO produk VALUES (3, 'Silver Queen', 25, 'pcs'); | | INSERT INTO produk VALUES (4, 'Pocky', 5, 'box'); |  1. Buatlah function yang akan digunakan oleh Trigger sebagai berikut:  |  | | --- | | -- Membuat Function untuk Trigger - INSERT, UPDATE, DELETE | | CREATE OR REPLACE FUNCTION fn\_update\_stok() | | RETURNS TRIGGER | | LANGUAGE PLPGSQL | | AS | | $$ | | BEGIN | | IF TG\_OP = 'INSERT' THEN | | UPDATE produk SET stok = (stok - NEW.jumlah) WHERE id\_produk = NEW.id\_produk; | | ELSIF TG\_OP = 'UPDATE' THEN | | UPDATE produk SET stok = ((stok + OLD.jumlah) - NEW.jumlah) WHERE id\_produk = OLD.id\_produk; | | ELSIF TG\_OP = 'DELETE' THEN | | UPDATE produk SET stok = (stok + OLD.jumlah) WHERE id\_produk = OLD.id\_produk; | | END IF; | |  | | RETURN NULL; | | END; | | $$   1. Buatlah Trigger yang akan memanggil function yang telah dibuat sebagai berikut:  |  | | --- | | -- Membuat Trigger yang memanggil function fn\_update\_stok() | | CREATE OR REPLACE TRIGGER trg\_update\_stok | | AFTER INSERT OR UPDATE OR DELETE on pesanan | | FOR EACH ROW | | EXECUTE FUNCTION fn\_update\_stok(); |  1. Buatlah contoh query untuk menguji operasi INSERT! 2. Buatlah contoh query untuk menguji operasi UPDATE! 3. Buatlah contoh query untuk menguji operasi DELETE! | |
| **Jawaban (Penjelasan dan Syntax)** |
| 1.a  Tabel produk menyimpan daftar barang dan jumlah stoknya. Tabel pesanan berisi data transaksi pembelian produk. Keduanya dihubungkan lewat id\_produk.  1.b  Function ini akan dipanggil otomatis oleh trigger untuk memperbarui stok.   * Jika ada pesanan baru (INSERT), stok berkurang. * Jika pesanan diubah (UPDATE), stok disesuaikan antara jumlah lama dan baru. * Jika pesanan dihapus (DELETE), stok dikembalikan.   1.c  Trigger ini akan aktif setiap kali ada perubahan data di tabel pesanan. Karena trigger dibuat AFTER, maka perubahan stok terjadi setelah operasi utama selesai.  1.d  menambah pesanan baru sebanyak **5 pcs** untuk produk **Choki-Choki** (id\_produk = 1). Trigger akan otomatis mengurangi stok produk tersebut.  **Syntax :**  INSERT INTO pesanan VALUES (101, 1, 5, CURRENT\_DATE);  1.e  Pesanan no. 101 diubah dari **5 pcs** menjadi **8 pcs**. Trigger akan menghitung ulang stok: stok baru = stok lama + jumlah lama - jumlah baru = 15 + 5 - 8 = **12**  **Syntax :**  UPDATE pesanan SET jumlah = 8 WHERE id\_pesanan = 101;  1.f  Pesanan no. 101 dihapus. Trigger otomatis menambahkan kembali stok sebanyak jumlah yang dihapus (8 pcs).  **Syntax :**  DELETE FROM pesanan WHERE id\_pesanan = 101; |
| **Hasil Eksekusi (Screenshot)** |
| 1.a      1.b      1.c |

# Soal No. 2

|  |
| --- |
| **Soal** |
| **Business Process #2**  A close up of a map  Description automatically generated  Pada Business Process #1, masih terdapat masalah dalam hal pemeriksaan ketersediaan stok yang dapat bernilai kurang dari nol (minus). Maka, diperlukan solusi untuk menjaga ketersediaan stok! Sehingga, diperlukan Business Process #2 untuk menyelesaikan masalah tersebut.   1. Buatlah **solusi untuk semua operasi** dalam menyelesaikan masalah di atas menggunakan CONSTRAINT CHECK! 2. Lakukan **pengujian** **untuk semua operasi** atas penerapan solusi menggunakan CONSTRAINT CHECK! 3. Buatlah **solusi untuk semua operasi** dalam menyelesaikan masalah di atas menggunakan conditional (IF-ELSE) dan menampilkan pesan error ketika stok tidak tersedia! 4. Lakukan **pengujian** **untuk semua operasi** atas penerapan solusi menggunakan conditional (IF-ELSE) dan menampilkan pesan error ketika stok tidak tersedia! |
| **Jawaban (Penjelasan dan Syntax)** |
| 2.a  **Penjelasan:**  CHECK digunakan agar kolom stok pada tabel produk tidak pernah bisa memiliki nilai di bawah 0. Jadi kalau ada operasi (trigger) yang menyebabkan stok jadi negatif, database akan otomatis menolak dan memunculkan error.  **Syntax :**  ALTER TABLE produk  ADD CONSTRAINT chk\_stok\_non\_negatif CHECK (stok >= 0);  2.b  **Uji Coba 1 (valid – stok cukup):**  INSERT INTO pesanan VALUES (201, 2, 5, CURRENT\_DATE);  **Output:**  INSERT 0 1  **Penjelasan:** stok awal Potato Stick = 15 → 10, masih valid (>=0).  **Uji Coba 2 (tidak valid – stok jadi negatif):**  INSERT INTO pesanan VALUES (202, 4, 10, CURRENT\_DATE);  **Output error dari PostgreSQL:**  ERROR: new row for relation "produk" violates check constraint "chk\_stok\_non\_negatif"  DETAIL: Failing row contains (4, Pocky, -5, box).  **Penjelasan:** stok Pocky cuma 5 box, tapi dipesan 10, sehingga hasil perhitungan (5–10 = -5) tidak lolos CHECK (stok >= 0) dan sistem menolak transaksi.  **Kelemahan CHECK :**  CHECK hanya bekerja setelah trigger dijalankan, jadi database akan memunculkan error setelah stok berubah negatif, bukan mencegah perhitungan dari awal. Untuk validasi yang lebih “cerdas” (misalnya langsung menolak sebelum stok berkurang), kita perlu pakai IF–ELSE di function trigger  2.c  Saya buat function baru yang memeriksa stok **sebelum** update terjadi. Jika stok tidak cukup, maka trigger akan menghentikan eksekusi dan memunculkan error dengan pesan yang kita tentukan sendiri  **Syntax :**  CREATE OR REPLACE FUNCTION fn\_update\_stok\_validasi()  RETURNS TRIGGER  LANGUAGE plpgsql  AS $$  DECLARE  stok\_saat\_ini NUMERIC;  BEGIN  IF TG\_OP = 'INSERT' THEN  SELECT stok INTO stok\_saat\_ini FROM produk WHERE id\_produk = NEW.id\_produk;  IF stok\_saat\_ini < NEW.jumlah THEN  RAISE EXCEPTION 'Stok produk tidak mencukupi! (tersisa: %, diminta: %)', stok\_saat\_ini, NEW.jumlah;  END IF;  UPDATE produk SET stok = stok - NEW.jumlah WHERE id\_produk = NEW.id\_produk;  ELSIF TG\_OP = 'UPDATE' THEN  SELECT stok INTO stok\_saat\_ini FROM produk WHERE id\_produk = NEW.id\_produk;  IF (stok\_saat\_ini + OLD.jumlah) < NEW.jumlah THEN  RAISE EXCEPTION 'Stok tidak cukup untuk update pesanan!';  END IF;  UPDATE produk SET stok = stok + OLD.jumlah - NEW.jumlah WHERE id\_produk = NEW.id\_produk;  ELSIF TG\_OP = 'DELETE' THEN  UPDATE produk SET stok = stok + OLD.jumlah WHERE id\_produk = OLD.id\_produk;  END IF;  RETURN NULL;  END;  $$;  **Membuat Trigger baru untuk function tersebut :**  CREATE OR REPLACE TRIGGER trg\_update\_stok\_validasi  BEFORE INSERT OR UPDATE OR DELETE  ON pesanan  FOR EACH ROW  EXECUTE FUNCTION fn\_update\_stok\_validasi();  2.d  **Uji Coba dan Hasil**  **1. INSERT pesanan valid (stok cukup):**  INSERT INTO pesanan VALUES (203, 3, 5, CURRENT\_DATE);  **Penjelasan:** Produk "Silver Queen" stok 25 → berkurang 5 jadi 20. Tidak ada error  **2. INSERT pesanan tidak valid (stok kurang):**  INSERT INTO pesanan VALUES (204, 4, 50, CURRENT\_DATE);  **Output error (custom dari trigger):**  ERROR: Stok produk tidak mencukupi! (tersisa: 5, diminta: 50)  CONTEXT: PL/pgSQL function fn\_update\_stok\_validasi() line 8 at RAISE  **Penjelasan:** Trigger menghentikan proses sebelum data masuk, karena stok tidak cukup. Data pesanan **tidak jadi ditambahkan**, dan stok produk tetap sama.  **3. UPDATE pesanan ke jumlah terlalu besar:**  UPDATE pesanan SET jumlah = 30 WHERE id\_pesanan = 203;  **Output:**  ERROR: Stok tidak cukup untuk update pesanan!  CONTEXT: PL/pgSQL function fn\_update\_stok\_validasi() line 14 at RAISE  **Penjelasan:** Trigger menolak update karena stok yang tersedia tidak mencukupi jumlah baru.  **4. DELETE pesanan:**  DELETE FROM pesanan WHERE id\_pesanan = 203;  **Output:**  DELETE 1  **Penjelasan:** Stok kembali bertambah sesuai jumlah yang dihapus. |
| **Hasil Eksekusi (Screenshot)** |
| 2.a    2.b  **valid**    **Tidak valid**    2.c      2.d  1.    2.    3.    4. |

# Soal No. 3

|  |
| --- |
| **Soal** |
| 1. Apakah Trigger dapat memanggil Function? Sertakan referensinya! 2. Apakah Trigger dapat memanggil Procedure? Lakukan percobaan untuk mengujinya langsung, dan sertakan referensinya! |
| **Jawaban (Penjelasan dan Syntax)** |
| 3.a. **Apakah Trigger dapat memanggil Function?**  **Jawaban:** Ya. **Penjelasan:** Ini adalah mekanisme standar di PostgreSQL. Trigger *harus* memanggil function yang RETURNS TRIGGER. **Code Syntax (Contoh):** Code syntax untuk ini sudah disediakan dan berhasil dijalankan pada **Soal No. 1**  **Referensi:**   * **Referensi Internal (Soal No. 1):** Pengerjaan pada Soal No. 1 di dokumen ini adalah bukti langsung. Trigger trg\_update\_stok dibuat menggunakan sintaks EXECUTE FUNCTION fn\_update\_stok();. * **Referensi Eksternal:** Dokumentasi resmi PostgreSQL dan referensi dari Neon secara eksplisit menyatakan sintaks CREATE TRIGGER ... EXECUTE FUNCTION function\_name().   **Syntax :**  -- Function dari Soal 1 (Contoh Function)  CREATE OR REPLACE FUNCTION fn\_update\_stok()  RETURNS TRIGGER  LANGUAGE PLPGSQL  AS  $$  BEGIN  RETURN NULL;  END;  $$;  -- Trigger dari Soal 1 (Memanggil Function)  CREATE OR REPLACE TRIGGER trg\_update\_stok  AFTER INSERT OR UPDATE OR DELETE on pesanan  FOR EACH ROW  EXECUTE FUNCTION fn\_update\_stok();  3b.  **Langkah 1: Membuat Procedure (Ini akan SUKSES)** Pertama, buat sebuah PROCEDURE sederhana  **Syntax :**  -- Membuat PROCEDURE (bukan function)  CREATE OR REPLACE PROCEDURE p\_test\_trigger()  LANGUAGE PLPGSQL  AS $$  BEGIN  -- Isi prosedur tidak penting,  -- ini hanya untuk membuktikan ia ada sebagai 'PROCEDURE'  END;  $$;  **Langkah 2: Percobaan membuat Trigger memanggil Procedure (Ini akan GAGAL)** Selanjutnya, coba buat TRIGGER yang memanggil PROCEDURE tersebut.  Sintaks trigger yang valid adalah EXECUTE FUNCTION. Saat kita mencoba mengarahkan sintaks valid tersebut ke nama PROCEDURE (p\_test\_trigger), Postgres akan mencari sebuah FUNCTION dengan nama itu.  **Syntax :**  -- Percobaan membuat TRIGGER (Ini akan GAGAL)  CREATE OR REPLACE TRIGGER trg\_test\_procedure  AFTER INSERT ON pesanan  FOR EACH ROW  EXECUTE FUNCTION p\_test\_trigger();  **Penjelasan Error (SQL state: 42883)**  Saat menjalankan Langkah 2 (CREATE TRIGGER), eksekusi akan gagal dan menghasilkan error: ERROR: function p\_test\_trigger() does not exist  Ini membuktikan bahwa:   1. Trigger menggunakan sintaks EXECUTE FUNCTION. 2. Postgres *hanya* mencari objek bertipe FUNCTION untuk dieksekusi. 3. Meskipun ada PROCEDURE bernama p\_test\_trigger, Postgres tidak dapat "melihat" atau menggunakannya sebagai FUNCTION, sehingga menghasilkan error "does not exist". |
| **Hasil Eksekusi (Screenshot)** |
| 3.a    3b.  Pembuktian sukses    Pembuktian gagal |