Bab-7

Triggers

Pokok Bahasan

- Membuat triggers

Tujuan Praktikum

Setelah melakukan praktikum mahasiswa diharapkan:

- Mengetahui dan mampu membuat trigger pada event sebuah table

Tugas Pendahuluan

- Jelaskan apa yang dimaksud dengan triggers
- Jelaskan keuntungan dan kerugian dari triggers

Percobaan 1 : Skema Table

1. Buat database dbsales
2. Buat table employee dengan perintah berikut:

```
dbsales=>CREATE TABLE employees(
   id serial primary key,
   first_name varchar(40) NOT NULL,
   last_name varchar(40) NOT NULL
);
```

3. Buat table employee_audits dengan perintah berikut:

```
dbsales=>CREATE TABLE employee_audits (
   id SERIAL PRIMARY KEY,
   employee_id int4 NOT NULL,
   last_name varchar(40) NOT NULL,
   changed_on timestamp(6) NOT NULL
);
```

Percobaan 2 : Buat Triggers

1. Buat fungsi dengan nama log_last_name_changes()

```
dbsales=>CREATE OR REPLACE FUNCTION log_last_name_changes()
  RETURNS trigger AS
$BODY$
BEGIN
    IF NEW.last_name <> OLD.last_name THEN
       INSERT INTO employee_audits(employee_id,last_name,changed_on)
```

```
        VALUES(OLD.id,OLD.last_name,now());
    END IF;
    RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;
```

2. Buat trigger yang akan memanggil fungsi log_last_name_changes()

```
dbsales=>CREATE TRIGGER last_name_changes
  BEFORE UPDATE
  ON employees
  FOR EACH ROW
  EXECUTE PROCEDURE log_last_name_changes();
```

Percobaan 3 : Studi kasus transaksi sales

1. Buat tabel transaksi dengan skema berikut ini: **(field id SERIAL, sales_id foreign key ke table employees)**

```
dbsales=# \d transaksi
                            Table "public.transaksi"
  Column  |       Type       |                       Modifiers
----------+------------------+-------------------------------------
 id       | integer          | not null default nextval('xxx)
 tanggal  | date             | default ('now'::text)::date
 jumlah   | double precision |
 sales_id | integer          |
Indexes:
    "transaksi_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "transaksi_sales_id_fkey" FOREIGN KEY (sales_id) REFERENCES
employees(id)
```

2. Tambahkan field persen_fee dan total_fee dengan tipe data double precision pada table employees dan update datanya menjadi seperti berikut ini

```
dbsales=# select * from employees;
 id | first_name | last_name | persen_fee | total_fee
----+------------+-----------+------------+-----------
  2 | Lily       | Brown     |       0.15 |         0
  1 | John       | Doe       |        0.2 |         0
```

3. Buat fungsi untuk update total_fee untuk setiap transaksi yang pernah dilakukan oleh sales

```
CREATE OR REPLACE FUNCTION update_feesales()
RETURNS TRIGGER AS
$$
  DECLARE
  BEGIN
    UPDATE employees SET total_fee=total_fee +
```

```
(persen_fee*NEW.jumlah)
                              WHERE id=NEW.sales_id;
    RETURN NEW;
  END;
$$
  language plpgsql;
```

4. Buat trigger yang akan menjalankan fungsi update_feesales() ketika data baru dimasukan ke table transaksi

```
CREATE TRIGGER trig_update_fee
AFTER INSERT OR UPDATE ON transaksi
FOR EACH ROW
EXECUTE PROCEDURE update_feesales();
```

5. Insert data ke table transaksi !!

```
dbsales=# insert into transaksi (jumlah,sales_id) values (20000,1);
dbsales=# insert into transaksi (jumlah,sales_id) values (500000,1);
dbsales=# insert into transaksi (jumlah,sales_id) values (2500000,2);
```

6. Tampilkan data hasil transaksi !!
7. Tampilkan data employees, apakah total_fee telah terupdate !!