Laporan Praktikum
Triggers
Database PostgreSQL

Ahmad Fathan Syakir
0110217055
Teknik Informatika 1

Sekolah Tinggi Teknologi Terpadu Nurul Fikiri
2018

Tugas Pendahuluan :

- Jelaskan apa yang dimaksud dengan triggers

Trigger adalah sekumpulan perintah untuk menjalan operasi tertentu seperti Insert, Update, Delete pada table atau view secara otomatis

- Jelaskan keuntungan dan kerugian dari triggers

keuntungan :
Trigger menyediakan cara alternative untuk memeriksa integritas, dan menyediakan cara alternative untuk menjalakan tugas-tugas yang dijadwalkan, dan trigger sangan berguna untuk mengaudit perubahan dalam table database.

Kekurangan :
Trigger hanya menyediakan validasi tambahan tapi tidak dapat menggantikan semua validasi, sulit mencari tahu apa yang terjadi pada level database karena trigger mengeksekusi secara tak terlihat dari klien-aplikasi yang terhubung pada database. Dan trigger berjalan setiap update yang dibuat ke table sehingga memberatkan beban pekerjaan pada database dan membuat system menjadi lambat

Percobaan 1 : Skema Table

1. Buat database “seles”

```
fatan@X441U:~$ /home/fatan/pg965/bin/createdb dbseles -U fatan -p5555 -h
localhost
Password:
fatan@X441U:~$ /home/fatan/pg965/bin/psql dbseles -U fatan -p5555 -h
localhost
Password for user fatan:
```

2. Buat table employee

```
dbseles=# create table employees(
dbseles(# id serial primary key,
dbseles(# first_name varchar(40) not null,
dbseles(# last_name varchar(40) not null)
dbseles-# ;
CREATE TABLE
```

3. Buat table employee_audits

```
dbseles=# create table employees_audits(
id serial primary key,
employee_id int4 not null,
last_name varchar(40) not null,
change_on timestamp(6) not null);
CREATE TABLE
```

Percobaan 2 : Buat Triggers

1. Buat fungsi dengan nama log_last_name_changes()

```
dbseles=# create or replace function log_last_name_changes() returns
trigger as
dbseles-# $body$
dbseles$# begin
dbseles$# if new.last_name <> old.last_name then
dbseles$# insert into employees_audits(employee_id, last_name, change_on)
dbseles$# values (old.id, old.last_name,now());
dbseles$# end if;
dbseles$# return new;
dbseles$# end;
dbseles$# $body$
dbseles-# language plpgsql;
CREATE FUNCTION
```

2. Buat trigger yang akan memanggil fungsi log_last_name_changes()

```
dbseles=# create trigger last_name_changes
dbseles-# before update
dbseles-# on employees
dbseles-# for each row
dbseles-# execute procedure log_last_name_changes();
CREATE TRIGGER
```

note : fungsi before/after adalah untuk menentukan kapan proses secara otomatis akan dieksekusi, sebelum atau sesudah proses.

3. Menampilkan table dan triggers yang sudah dibuat

```
dbseles=# \d
                  List of relations
 Schema |           Name            |   Type   | Owner
--------+---------------------------+----------+-------
 public | employees                 | table    | fatan
 public | employees_audits          | table    | fatan
 public | employees_audits_id_seq   | sequence | fatan
 public | employees_id_seq          | sequence | fatan
(4 rows)

dbseles=# \df
                                List of functions
 Schema |         Name          | Result data type | Argument data types |  Type
--------+-----------------------+------------------+---------------------+---------
 public | log_last_name_changes | trigger          |                     | trigger
(1 row)
```

Percobaan 3 : Studi kasus transaksi sales

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

```
dbseles=# create table transaksi(
dbseles(# id serial primary key,
dbseles(# tanggal date,
dbseles(# jumlah double precision,
dbseles(# seles_id integer references employees(id));
CREATE TABLE
dbseles=# select * from transaksi;
 id | tanggal | jumlah | seles_id
----+---------+--------+----------
(0 rows)
```

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

```
dbseles=# alter table employees add persem_fee double precision;
ALTER TABLE

dbseles=# alter table employees add total_fee double precision;
ALTER TABLE

dbseles=# select * from employees;
 id | first_name | last_name | persem_fee | total_fee
----+------------+-----------+------------+-----------
  1 | Ahmad      | Fathan    |            |
  2 | Muhammad   | Akbar     |            |
  3 | raihan     | rabbani   |            |
(3 rows)
```

note : menambahkan field baru yaitu persem_fee dan total_fee

```
dbseles=# update employees set persem_fee = '0.2' where id = 1;
UPDATE 1

dbseles=# update employees set persem_fee = '0.15' where id = 2;
UPDATE 1

dbseles=# update employees set persem_fee = 0.5 where id = 3;
UPDATE 1

dbseles=# select * from employees;
 id | first_name | last_name | persem_fee | total_fee
----+------------+-----------+------------+-----------
  1 | Ahmad      | Fathan    |        0.2 |
  2 | Muhammad   | Akbar     |       0.15 |
  3 | raihan     | rabbani   |        0.5 |
(3 rows)
```

note : mengisi data pada field persem_fee

```
dbseles=# update employees set total_fee = '0' where id = 3;
UPDATE 1

dbseles=# update employees set total_fee = '0' where id = 2;
UPDATE 1

dbseles=# update employees set total_fee = '0' where id = 1;
UPDATE 1

dbseles=# select * from employees;
 id | first_name | last_name | persem_fee | total_fee
----+------------+-----------+------------+-----------
  1 | Ahmad      | Fathan    |        0.2 |           0
  2 | Muhammad   | Akbar     |       0.15 |           0
  3 | raihan     | rabbani   |        0.5 |           0
(3 rows)
```

note : mengisi data pada field total_fee agar saat triggers bisa berjalan karna pada triggers yang dijalan perintah update bukan insert

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

```
dbseles# create or replace function update_feeseles() returns trigger as
$body$
declare
begin
update employees set total_fee = total_fee + (persem_fee*new.jumlah)
where id = new.seles_id;
return new;
end;
$body$
language plpgsql;
CREATE FUNCTION
```

note : function ini digunakan menghitung data pada field jumlah pada table transaksi (yang diupdate) dan menghitungnya dengan field persem_fee pada table employee

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

```
dbseles=# create trigger trig_update_fee
after insert or update on transaksi
for each row
execute procedure update_feeseles();
CREATE TRIGGER
```

note : trigger ini digunakan untuk otomatis mengupdate data pada table employee yang dimana data tersebut diambil dari function update_feeseles yang dibuat sebelumnya

5. Insert data ke table transaksi

```
dbseles=# insert into transaksi (id, tanggal, jumlah, seles_id) values
(1, '20181010', 0, 1), (2, '20181018', 0, 2), (3, '20181020', 0, 3);
INSERT 0 3
dbseles=# select * from transaksi;
 id |  tanggal   | jumlah | seles_id
----+------------+--------+----------
  1 | 2018-10-10 |      0 |        1
  2 | 2018-10-18 |      0 |        2
  3 | 2018-10-20 |      0 |        3
(3 rows)

dbseles=# update transaksi set jumlah = 20000 where id = 1;
UPDATE 1
dbseles=# update transaksi set jumlah = 2500000 where id = 2;
UPDATE 1
dbseles=# update transaksi set jumlah = 300000 where id = 3;
UPDATE 1
```

note : table ini digunakan untuk menghitung function yang dibuat lalu dikirim ke table employee menggunakan trigger

6. Tampilkan data hasil transaksi

```
dbseles=# select * from transaksi;
 id |  tanggal   | jumlah  | seles_id
----+------------+---------+----------
  1 | 2018-10-10 |   20000 |        1
  2 | 2018-10-18 | 2500000 |        2
  3 | 2018-10-20 |  300000 |        3
(3 rows)
```

7. Tampilkan data employee, apakah total_fee telah terupdate

```
dbseles=# select * from employees;
 id | first_name | last_name | persem_fee | total_fee
----+------------+-----------+------------+-----------
  1 | Ahmad      | Fathan    |        0.2 |      4000
  2 | Muhammad   | Akbar     |       0.15 |    375000
  3 | raihan     | rabbani   |        0.5 |    150000
(3 rows)
```

note : hasil dari perhitungan function update_feeseles