



Program Studi : Teknik Informatika

Laporan Praktikum : Basis Data 2

Praktikum 5

Muhammad Azhar Rasyad
0110217029

**STT Terpadu Nurul Fikri
Tahun 2018**

Triggers

Tugas Pendahuluan

1. Jelaskan apa yang dimaksud dengan triggers !

Triggers adalah aksi yang dilakukan jika terjadi perubahan pada row data suatu table.

2. Jelaskan keuntungan dan kerugian dari triggers !

Keuntungan Triggers :

- Trigger menyediakan cara alternative untuk memeriksa integritas.
- Trigger bisa menangkap kesalahan dalam business logic pada tingkat database.
- Trigger menyediakan cara alternative untuk menjalankan tugas-tugas yang dijadwalkan.
- Trigger sangat berguna untuk mengaudit perubahan data dalam table database

Kerugian Triggers :

- Trigger hanya bisa menyediakan validasi tambahan tapi tidak dapat menggantikan semua validasi.
- Beberapa validasi sederhana dapat dilakukan di level aplikasi.
- Trigger mengeksekusi secara tak terlihat dari klien-aplikasi yang terhubung ke database server sehingga sulit untuk mencari tahu apa yang terjadi di level database.
- Trigger berjalan setiap update yang dibuat ke table sehingga menambah beban kerja ke database dan menyebabkan system berjalan lebih lambat.

Sumber :

http://apanyaa.blogspot.com/2012/03/laporan-matakuliah-pengolahan-basis_15.html

Percobaan 1 : Skema Table

1. Buat database dbsales

```
apsql@mazharrasyad: ~
mzharrasyad@mzharrasyad:~$ sudo su - apsql
[sudo] password for mzharrasyad:
apsql@mzharrasyad:~$

apsql@mzharrasyad:~$ /home/apsql/pg105/bin/pg_ctl -D /home/apsql/datapg/ -l log
file start
waiting for server to start.... done
server started
apsql@mzharrasyad:~$

apsql@mzharrasyad:~$ /home/apsql/pg105/bin/createdb dbsales -U apsql -p5555 -h
localhost
Password:
apsql@mzharrasyad:~$

apsql@mzharrasyad:~$ /home/apsql/pg105/bin/psql dbsales -U apsql -p5555 -h loca
localhost
Password for user apsql:
psql (10.5)
Type "help" for help.

dbsales=#
```

2. Buat table employees dengan perintah berikut :

```
apsql@mazharrasyad: ~
GNU nano 2.5.3      File: /tmp/psql.edit.13495.sql      Modified
create table employees(
id serial primary key,
first_name varchar(40) not null,
last_name varchar(40) not null
);
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Uncut Text  ^T To Spell     ^_ Go To Line
```

3. Buat table employees_audits dengan perintah berikut :

```
apsql@mazharrasyad: ~
GNU nano 2.5.3      File: /tmp/psql.edit.13495.sql      Modified

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
);
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

4. List table :

```
apsql@mazharrasyad: ~
dbsales=# \d

          List of relations
Schema |          Name          |  Type   | Owner
-----+-----+-----+-----
public | employees              | table   | apsql
public | employees_audits       | table   | apsql
public | employees_audits_id_seq | sequence | apsql
public | employees_id_seq       | sequence | apsql
(4 rows)
```

dbsales=#

Percobaan 2 : Buat Triggers

1. Buat fungsi dengan nama `log_last_name_changes()`

```

GNU nano 2.5.3 File: /tmp/psql.edit.13495.sql Modified
create or replace function
Log_last_name_changes() returns trigger as
$body$
begin
    if new.last_name <> old.last_name then
        insert into employees_audits(employee_id,last_name,change_on)
        values(old.id,old.last_name,now());
    end if;
    return new;
end
$body$ language plpgsql;

```

^G Get Help
^O Write Out
^W Where Is
^K Cut Text
^J Justify
^C Cur Pos
^X Exit
^R Read File
^_ Replace
^U Uncut Text
^T To Spell
^_ Go To Line

2. Buat trigger yang akan memanggil fungsi log_last_name_changes()

```

GNU nano 2.5.3      File: /tmp/psql.edit.13495.sql      Modified
create trigger last_name_changes before
update on employees for each row
execute procedure log_last_name_changes();

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

- ### 3. List Function

```

apssl@mazharryad: ~
dbsales=# \df

```

List of functions				
Schema	Name	Result data type	Argument data types	Type
public	log_last_name_changes	trigger		trigger
public	update_feesales	trigger		trigger

```

(2 rows)

dbsales=#

```

4. Menambahkan data table employees

```
apsql@mazharrasyad: ~
GNU nano 2.5.3      File: /tmp/psql.edit.3049.sql      Modified
insert into employees (id,first_name,last_name) values
(1,'John','Doe'),
(2,'Lily','Brow');
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

5. Menampilkan data table employees

```
apsql@mazharrasyad: ~
dbsales=# select * from employees;
 id | first_name | last_name
-----+-----+-----
  1 | Lily       | Brow
  2 | John       | Doe
(2 rows)

dbsales=#
```

6. Mengubah data Brown menjadi Brown

```
apsql@mazharrasyad: ~
dbsales=# update employees set last_name='Brown' where id=2;
UPDATE 1
dbsales=#
```

7. Tampilan setelah diubah

```
apsql@mazharrasyad: ~
dbsales=# select * from employees;
 id | first_name | last_name
-----+-----+-----
  1 | John       | Doe
  2 | Lily       | Brown
(2 rows)

dbsales=#
```

8. Tampilan log setelah diupdate

```
apsql@mazharrasyad: ~
dbsales=# select * from employees_audits;
 id | employee_id | last_name | change_on
-----+-----+-----+-----
  1 |          2 | Brow     | 2018-10-15 12:51:19.768494
(1 row)

dbsales=#
```

Percobaan 3 : Studi kasus transaksi sales

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

```
apsql@mazharrasyad: ~
GNU nano 2.5.3      File: /tmp/psql.edit.3049.sql      Modified

create table transaksi(
id serial primary key,
tanggal date default now(),
jumlah double precision,
sales_id integer references employees(id)
);

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

```
apsql@mazharrasyad: ~
dbsales=# \d transaksi

              Table "public.transaksi"
   Column   |      Type      | Collation | Nullable |      Default
-----+-----+-----+-----+-----
 id         | integer        |           | not null | nextval('transaksi_id_seq'::regclass)
 tanggal   | 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)
Triggers:
    trig_update_fee AFTER INSERT OR UPDATE ON transaksi FOR EACH ROW EXECUTE PROCEDURE upda
te_feesaes()
dbsales=#
```

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

```
apsql@mazharrasyad: ~
dbsales=# alter table employees add persen_fee double precision;
ALTER TABLE
dbsales=# alter table employees add total_fee double precision;
ALTER TABLE
dbsales=#
```

```
apsql@mazharrasyad: ~
dbsales=# select * from employees;
 id | first_name | last_name | persen_fee | total_fee
-----+-----+-----+-----+-----
  1 | John       | Doe       |           | 
  2 | Lily       | Brown     |           | 
(2 rows)

dbsales=#
```

```
GNU nano 2.5.3      File: /tmp/psql.edit.3049.sql      Modified
update employees set persen_fee=0.2 where id=1;
update employees set persen_fee=0.15 where id=2;
update employees set total_fee=0 where id in(1, 2);

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To Line
```

```
apsql@mazharrasyad: ~
dbsales=# select * from employees;
 id | first_name | last_name | persen_fee | total_fee
-----+-----+-----+-----+-----
  1 | John       | Doe       |         0.2 |         0
  2 | Lily       | Brown     |         0.15 |         0
(2 rows)

dbsales=#
```

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

```
GNU nano 2.5.3      File: /tmp/psql.edit.15952.sql      Modified
create or replace function
update_feessales() returns trigger as
$body$
begin
    update employees set total_fee = total_fee + (persen_fee * new.jumlah) where id = new.sales_id;
    return new;
end
$body$ language plpgsql;
```


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

```
apsql@mazharrasyad: ~
GNU nano 2.5.3      File: /tmp/psql.edit.15952.sql      Modified
create trigger trig_update_fee after
insert or update on transaksi for each row
execute procedure update_feesales();
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

5. Insert data ke table transaksi !!

```
apsql@mazharrasyad: ~
GNU nano 2.5.3      File: /tmp/psql.edit.3049.sql      Modified
insert into transaksi (jumlah,sales_id) values
(20000,1),
(500000,1),
(2500000,2);
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

6. Tampilkan data hasil transaksi !!

```
apsql@mazharrasyad: ~
dbsales=# select * from transaksi;
 id | tanggal   | jumlah  | sales_id
-----+-----+-----+-----
  1 | 2018-10-15 |  20000 |         1
  2 | 2018-10-15 | 500000 |         1
  3 | 2018-10-15 |2500000 |         2
(3 rows)

dbsales=#
```

7. Tampilkan data employees, apakah total_fee telah terupdate !!

```
apsql@mazharrasyad: ~
dbsales=# select * from employees;
 id | first_name | last_name | persen_fee | total_fee
-----+-----+-----+-----+-----
  1 | John       | Doe       |         0.2 | 104000
  2 | Lily       | Brown     |         0.15 | 375000
(2 rows)

dbsales=#
```

----- Selesai -----