BAB VIII
FUNGSI, PL/PGSQL dan TRIGER

8.1 Bahasan dan Sasaran

8.1.1 Bahasan

- Pada bab kali ini akan membahas tentang fungsi
- Selain hal itu akan dibahas juga mengenai pl/pgsql dan triger

8.1.2 Sasaran

- Mahasiswa memahami dan menggunakan fungsi dalam database PostgreSQL
- Mahasiswa memahami dasar penggunaan PL/PGSQL dan TRIGER

8.2 Materi

8.2.1 Pendukung Fungsi

PostgreSQL memiliki fungsi yang dapat mengubah suatu nilai dalam suatu kolom atau barismenjadi huruf besar. Fungsi tersebut bernama *upper(nama_kolom)*, berfungsi memanggilfungsi *upper* dengan *nama_kolom* sebagaii argumen sehingga menghasilkan nilai padakolom dalam huruf besar. Berikut Struktur SQL untuk menampilkan data dalam huruf besar semua :

```
SELECT upper ([nama kolom]) FROM [nama tabel];
```

Berikut struktur SQL untuk menampilkan data dimana huruf pertama saja yang besar :

```
SELECT initcap ([nama kolom]) FROM [nama tabel];
```

8.2.2 Fungsi

Fungsi SQL adalah sebuah kumpulan query, biasanya query yang detail dan panjang yang dibungkus menjadi satu dan disimpan dalam *database* dan kemudian apabila diperlukan hanya tinggal mengaksesnya tanpa mengetikan query detail. Sedangkan untuk memunculkan fungsi dapat menggunakan query \df.

Ada beberapa konsep yang menarik dari fungsi antara lain:

- Bahasa yang dipakai dapat didefenisikan sendiri dengan tersedianya parameter LANGUAGE, tanpa harus mengkompilasi ulang PostgreSQL.
- Kita dapat membuat dua buah fungsi dengan nama yang sama namun parametermasukkannya yang berbeda, baik tipe data maupun jumlahnya.

Ada beberapa faktor yang perlu diperhatikan dalam membuat fungsi antara lain:

• Nama Fungsi

• Nomor dari fungsi *argument*

• Tipe data dari setiap *argument*

• Tipe dari hasil fungsi

• Fungsi *action*

• Bahasa yang digunakan oleh fungsi *action.*

Berikut contoh sederhana pembuatan fungsi SQL untuk perkalian dari tiga inputan :

```
db_personal=> CREATE FUNCTION perkalian (FLOAT, FLOAT, FLOAT)
db_personal-> RETURNS FLOAT
db_personal-> AS 'SELECT ($1 + $2) * $3;'
db_personal-> LANGUAGE 'sql';
CREATE

db_personal=> SELECT perkalian (10,10,10);
perkalian
-----------
200
(1 row)
```

Contoh yang lain dalam pembuatan fungsi SQL untuk mencari jumlah pegawai dari tabel pegawai berikut :

ID	NAMA	ALAMAT	NO.TELP	JABATAN	Gaji
1	Hendro	Solo	081223300	Teknisi	900000
2	Tika	Semarang	0897735357	Sekretaris	2000000
3	Wijaya	Jogjakarta	0865433225	Kepala	3000000
4	Dodi	Banyuwangi	076544677	Teknisi	1000000

Berikut kode SQL nya :

```
Create function jumlah_pegawai()
Returns bigint
As ‘select count(*) as jumlah_pegawai from pegawai;’
Language ‘sql’;
```

Maka hasilnya sebagai berikut :

```
Select jumlah_pegawai();

Jumlah_pegawai
----------------------
4
(1 row)
```

8.2.3 Fungsi PL/PGSQL

PL/PGSQL merupakan bahasa yang lain untuk membuat sebuah fungsi, biasanya bahasa ini digunakan untuk menangani fungsi yang lebih kompleks. Pl/pgsql sudah terdapat dalam instalasi PostgreSQL.

Keuntungan penggunaan Fungsi PL/PGSQL :

1. Meningkatkan kinerja karena mengurangi pengiriman kode dari klien ke server.
2. Meningkatkan keamanan karena pengaksesan data tertentu ditangani dalam server.
3. Meningkatkan konsistensi data saat sejumlah aplikasi memanggil prosedur yang sama;

Sebaliknya kelemahannya yaitu server akan lebih terbebani karena banyak proses yang harus ditangani. Sedangkan query PL/PGSQL agar lebih mudah di pahami akan dibagi menjadi 2 yaitu pembuatan fungsi dan pembuatan definisi

- **Berikut Struktur pembuatan fungsi dalam pl/pgsql**

1. Pembuatan fungsi :

```
CREATE [OR REPLACE] FUNCTION nama_fungsi (  argtype , ... ])
RETURNS tipe_data
AS ‘definisi’
LANGUAGE ‘plpgsql’;
```

2. Pembuatan definisi :

```
DECLARE nama_variable tipe_data  /* deklarasi variabel, type */
BEGIN
/* prosedural dan SQL masuk disini seperti select, update dan sebagainya*/
Return nama_variable /* blok ini yang wajib */
END;
```

- **Menghapus fungsi :**

DROP FUNCTION nama_fungsi(paramater, parameter, parameter ...);

Contoh :

DROP FUNCTION pembagian(text);

Berikut ini implementasi dari contoh penggunaan fungsi dengan bahasa PL/PGSQL:

```
db_personal=> CREATE FUNCTION pl_caripegawai (integer)
db_personal-> RETURNS text
db_personal-> AS 'DECLARE hasil TEXT;
db_personal'> BEGIN
db_personal'> SELECT INTO hasil nama as nama_mahasiswa
db_personal'> FROM pegawai
db_personal'> WHERE id = $1;
db_personal'> RETURN hasil;
db_personal'> END;'
db_personal-> LANGUAGE 'plpgsql';
CREATE

db_personal=> SELECT pl_caripegawai (1);
pl_caripegawai
-----------------
Hendro
(1 row)
db_personal=> SELECT pl_caripegawai (2);
pl_caripegawai
--------------------
Tika
(1 row)
```

Contoh berikut menunjukkan query percabangan yang akan menampilkan nilai terkecil dari dua buah parameter :

```
create function percabangan (x integer,y integer)
returns integer
as ‘declare nilai_terkecil integer;
begin
if  x < y then
select into nilai_terkecil x;
else
select into nilai_terkecil y;
end if;
return nilai_terkecil;
end;’
language ‘plpgsql’;
```

Hasil sebagai berikut :

```
Select percabangan (300,250);

Percabangan
----------------
250
(1 row)
```

Contoh berikut menunjukkan query perulangan yang akan menampilkan akumulasi dari jumlah perulangan :

```
Create function perulangan (a integer,b integer)
Returns integer
As ‘ declare nilai_awal integer default a;
Hasil integer default 0;
Begin

Loop

If nilai_awal > b then
Exit;
End if;
Hasil := hasil+nilai_awal;
Nilai_awal := nilai_awal+1;

End loop;

Return hasil;
End;’
Language ‘plpgsql’;
```

Hasilnya sebagai berikut :

```
Select perulangan (1,5);
Perulangan
----------
15
1 (row)
```

8.2.4 Triger

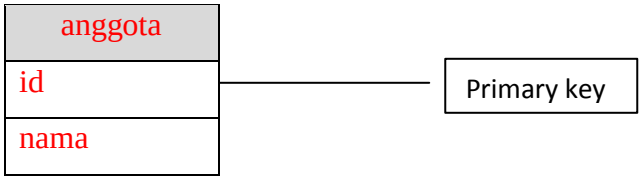
Trigger digunakan untuk menyisipkan sebuah fungsi pada saat suatu *record* di-INSERT, UPDATE dan DELETE. Trigger sangat ideal untuk mengecek atau memodifikasi sebuah data pada kolom sebelum dimasukkan ke dalam *database*, sehingga sebuah fungsi dapat dipanggil setiap saat secara otomatis ketika sebuah *row* akan dimodifikasi. Ciri khas dari fungsi yang diperuntukkan untuk *trigger* adalah menghasilkan output bertipe OPAQUE. Tipe *opaque* adalah sebuah tipe yang menginformasikan pada *database* bahwa fungsi tersebut tidak menghasilkan satu dari tipe data yang ditetapkan SQL dan tidak secara langsung dapat digunakan dalam statemen SQL. Language (bahasa) PL/PGSQL dapat digunakan untuk *trigger procedure*, fungsi untuk *trigger* ini memiliki beberapa variabel khusus yang terdeklarasi secara otomatis. Variabel tersebut antara lain:

• NEW: Variabel yang berisi nilai baru suatu record pada saat INSERT atau UPDATE, bertipe RECORD.

• OLD: Variabel yang berisi nilai lama suatu record pada saat UPDATE atau DELETE, juga bertipe RECORD.

Berikut ini beberapa contoh penggunaan fungsi sebagai *trigger procedure*:

Contoh : *trigger* berikut ini memastikan isi *field* atau kolom *nama* pada tabel *anggota* selalu huruf besar.



langkah pertama buatlah fungsinya terlebih dahulu :

```
db_personal=> CREATE FUNCTION tes_trigger()
db_personal-> RETURNS opaque
db_personal-> AS 'BEGIN
db_personal'> NEW.nama := UPPER(NEW.nama);
db_personal'> RETURN NEW;
db_personal'> END;'
db_personal-> LANGUAGE 'plpgsql';
CREATE
```

Kemudian lanjutkan dengan pembuatan *trigger* yang berfungsi untuk memanggil fungsi secara otomatis ketika kita melakukan INSERT ataupun UPDATE pada tabel *anggota*.

```
db_personal=> CREATE TRIGGER tes1_trigger
db_personal-> BEFORE INSERT
db_personal-> ON anggota
db_personal-> FOR EACH ROW
db_personal-> EXECUTE PROCEDURE tes_trigger();
CREATE
```

cobalah INSERT beberapa data ke dalam tabel anggota:

```
db_personal=> INSERT INTO anggota (id, nama)
db_personal-> VALUES (26, 'andhie');
INSERT 70831 1
db_personal=> INSERT INTO anggota
db_personal-> VALUES (83, 'rWatia');
INSERT 70832 1
```

tampilkan isi dari tabel *anggota*, hasilnya seperti pada tabel di bawah ini. Jadi setiap data yang kita INSERT walaupun dalam penulisannya menggunakan huruf kecil

namun secara otomatis *trigger* akan memanggil fungsi yang bertugas untuk mengganti setiap data yang masuk agar hasilnya nanti selalu menjadi huruf besar:

```
db_personal=> SELECT * FROM anggota;
id | nama
--------+-------------
26 | ANDHIE
83 | RWATIA
(2 rows)
```

Tugas Praktikum

1. Buatlah fungsi konversi suhu dari Fahrenheit ke derajat celcius dengan rumus konversi sebagai berikut : celcius= (5*(nilai Fahrenheit-32))/9!
   Kemudian jalankan hasilnya dengan menSELECT fungsi tersebut !
2. Buatlah fungsi untuk mencari alamat mahasiswa dari tabel mahasiswa berdasarkan nama mahasiswa. Kemudian jalankan dengan perintah SELECT !
3. Buatlah fungsi untuk menghitung nilai dengan menggunakan bahasa pl/sql !

Nilai > 100 atau Nilai<0	Nilai Salah
Nilai ≥ 90	Nilai A
70 ≤ Nilai < 90	Nilai B
60 ≤ Nilai < 70	Nilai C
50 ≤ Nilai < 60	Nilai D
0 ≤ Nilai < 50	Nilai E

   Kemudian jalankan hasilnya dengan menSELECT fungsi tersebut !
4. Buatlah fungsi menggunakan pl/pgsql untuk mencari bilangan ganjil atau genap dari bilangan yang diinputkan. Kemudian jalankan dengan perintah SELECT !
5. Tambahkan kolom modifikasi pada tabel mahasiswa. Dimana setiap ada insert atau update maka tanggal pada kolom modifikasi akan menunjukkan tanggal perubahan tersebut dilakukan.

Tugas Rumah

1. Buat laporan praktikum menggunakan DBMS mysql untuk mengerjakan tugas praktikum 1-5 dan letakkan di blog.