BASIS DATA II

Sirojul Munir | rojulman@nurulfikri.ac.id | @rojulman

Programming SQL
PL / PgSQL

Sirojul Munir | rojulman@nurulfikri.ac.id | @rojulman

Apa itu PL/pgSQL

- PL/pgSQL : Procedural Language/PostgreSQL)
- Procedural Language yang di support oleh PostgreSQL
- User dapat melalukan control data melalui program berbentuk procedural dibandingkan dengan perintah SQL biasa

PL/pgSQL :: Fitur

- Program procedural language dapat diload (dieksekusi) oleh PostgreSQL dengan fitur:
  - ✓ Dapat digunakan pada function dan prosedur trigger
  - ✓ Menambahkan struktur control pada perintah SQL
  - ✓ Dapat digunakan untuk perhitungan (komputasi) yang komplek
  - ✓ mewarisi semua : user-defined types, functions, and operators
  - ✓ didefinisikan pada server
  - ✓ mudah digunakan

PL/pgSQL :: Struktur

PL/pgSQL is a block-structured language.

PL/pgSQL Block Structure

```
[ <<label>> ]
[ DECLARE
    declarations ]
BEGIN
    statements
END [ label ];
```

Function :: Structure

- CREATE Function

```
CREATE FUNCTION identifier (arguments) RETURNS type AS '
  DECLARE
    declaration;
    [...]
  BEGIN
    statement;
    [...]
  END;
' LANGUAGE 'plpgsql';
```

Function :: Create

- Fungsi salam
-

```
CREATE FUNCTION salam()
RETURNS text AS
$$
 DECLARE
   -- deklarasi variabel say bertipe TEXT
   say TEXT;
 BEGIN
   -- inisialisasi variabel say
   say := 'Assalamualaikum Teman ';
   -- mengembalikan nilai fungsi
   RETURN say;
END;
$$  LANGUAGE plpgsql

select salam();
~
```

Function :: Run

- Melihat daftar fungsi (df) dan menjalankan fungsi (select function_name())
-

```
dblatihan=# \df
                                  List of functions
 Schema |    Name     | Result data type  |  Argument data types   |  Type
--------+-------------+-------------------+------------------------+--------
 public | hello       | character varying | nama character varying | normal
 public | hello_world | void              |                        | normal
 public | salam       | text              |                        | normal
(3 rows)

dblatihan=# select salam();
         salam
-------------------------
 Assalamualaikum Teman
(1 row)

dblatihan=# select hello('faiz');
   hello
------------
 Hello faiz
(1 row)
```

Function :: dengan argumen

- Fungsi menghitung luas_segitiga, dengan argumen 2 bilangan real dan nilai balik hasil hitung luas segitiga tipe data real
  - Argumen 1 : $1 -> alas segitiga
  - Argumen 2 : $2 -> tinggi segitiga
-

```
CREATE OR REPLACE FUNCTION luasSegiTiga(real , real) RETURNS real AS '
DECLARE
  alas ALIAS FOR $1;
  tinggi ALIAS FOR $2;
BEGIN
  RETURN alas * tinggi * 0.5;
END
' LANGUAGE plpgsql;
```

Function :: Run Query

```
latihan=> SELECT luasSegiTiga(4,7);
 luassegitiga
 --------------
           14
 (1 row)
```

Drop Function

```
dblatihan=> DROP FUNCTION hello() ;
   DROP FUNCTION
```

Soal

- ❑ Buat fungsi untuk menghitung jumlah dua bilangan
- ❑ Buat fungsi untuk menghitung luas lingkaran
- ❑ Buat fungsi untuk nilai rata dari total 3 nilai yang diberikan

Fungsi manipulasi String

- concatenation (penggabungan string) : SELECT ‘nurul ‘ || ‘fikri ‘ as nama
- panjang string : length(string)
- huruf besar : upper(string), huruf_kecil lower(string)
- huruf pertama kata besar : initcap(string)
- sub string : substring(‘Jakarta’, 1, 4) : jaka , substring(‘Jakarta’, 3, 5) : karta
- mengganti karakter string : replace(‘nurul’,’ul’,’il’) : nuril
- menghapus spasi kosong diawal & akhir string: trim(‘ nurul fikri ‘)
- ltrim / rtrim : menghapus karakter di sebelah kiri/kanan string
  - ltrim(‘z’,’znurulz’) : nurulz , rtrim(‘z’,’znurulz’) : znurul , trim(z,’znurulz’) : nurul
- menggabungkan karakter di sebelah kiri string: lpad(‘ul’,5,’nur’) : nurul

Fungsi waktu

- current_date() : tanggal sekarang
- current_time() : jam sekarang
- current_timestamp / now() : tanggal dan jam sekarang
- timeofday() : hari tanggal dan jam sekarang
- age(timestamp tgl_lahir, [timestamp tgl_sekarang]) : umur
- date_part('month', current_date) : bulan sekarang
- date_part('year', current_date) : tahun sekarang
- date_part('year', timestamp '1980-02-10') : 1980
- date_part('day', current_date) : tanggal sekarang

Fungsi to_char(nilai , format) : number ke string

Parameter	Explanation
9	Value (with no leading zeros)
0	Value (with leading zeros)
.	Decimal
,	Group separator
PR	Negative value in angle brackets
S	Sign
L	Currency symbol
D	Decimal
G	Group separator
MI	Minus sign (for negative numbers)
PL	Plus sign (for positive numbers)
SG	Plus/minus sign (for positive and negative numbers)
RN	Roman numerals
TH	Ordinal number suffix
th	Ordinal number suffix
V	Shift digits
EEEE	Scientific notation

Contoh:

- to_char(1501,'9999.99') : 1501.00
- to_char(1501,'9G999.99') : 1.501.00
- to_char(1501,'L9G999.99') : Rp 1.501.00
- to_char(201,’00999’) : 00201

Fungsi to_char(nilai , format) : date ke string

- YYYY : tahun :: 2015
- MM :: bulan :: 10 (oktober)
- DD :: tanggal :: 09
- HH :: jam : 10
- MI :: menit
- SS :: detik

Contoh:

```
select to_char(current_date,'dd-MM-YYYY') : 21-10-2015
select to_char(date '1945-08-17','dd-MM-YYYY') : 17-08-1945
```

Fungsi to_date(nilai , format) : string ke date

- YYYY : tahun :: 2015
- MM :: bulan :: 10 (oktober)
- DD :: tanggal :: 09
- HH :: jam : 10
- MI :: menit
- SS :: detik

Contoh:

```
select to_date('2010/09/20','YYYY/MM/DD') :2010-09-20
select to_date('101101','YYMMDD'): 2010-11-01
```

LIKE / ILIKE

- string LIKE/ILIKE pattern (LIKE : case sensitive , ILIKE : case unsensitive)
- string NOT LIKE/ILIKE pattern
  - `'abc' LIKE 'abc' true`
  - `'abc' LIKE 'a%' true`
  - `'abc' LIKE '_b_' true`
  - `'abc' LIKE 'c' false`
  - `'abc' LIKE 'A%' false`
  - `'abc' ILIKE 'A%' true`
  - `'abc' ILIKE '_B_' true`

SIMILAR TO

- ’abc’ SIMILAR TO ’abc’ true
- ’abc’ SIMILAR TO ’a’ false
- ’abc’ SIMILAR TO ’%(b|d)%’ true
- ’abc’ SIMILAR TO ’(b|c)%’ false

Operator Pembanding

Operator	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
=	equal
<> or !=	not equal

Operator Matematik

Operator	Description	Example	Result
+	addition	2 + 3	5
-	subtraction	2 - 3	-1
*	multiplication	2 * 3	6
/	division (integer division truncates the result)	4 / 2	2
%	modulo (remainder)	5 % 4	1
^	exponentiation	2.0 ^ 3.0	8
|/	square root	|/ 25.0	5
||/	cube root	||/ 27.0	3
!	factorial	5 !	120
!!	factorial (prefix operator)	!! 5	120
@	absolute value	@ -5.0	5
&	bitwise AND	91 & 15	11
|	bitwise OR	32 | 3	35
#	bitwise XOR	17 # 5	20
~	bitwise NOT	~1	-2
<<	bitwise shift left	1 << 4	16
>>	bitwise shift right	8 >> 2	2

Fungsi Matematika (1)

Function	Return Type	Description	Example	Result
abs(x)	(same as input)	absolute value	abs(-17.4)	17.4
cbrt(dp)	dp	cube root	cbrt(27.0)	3
ceil(dp or numeric)	(same as input)	smallest integer not less than argument	ceil(-42.8)	-42
ceiling(dp or numeric)	(same as input)	smallest integer not less than argument (alias for ceil)	ceiling(-95.3)	-95
degrees(dp)	dp	radians to degrees	degrees(0.5)	28.6478897565412
div(y numeric, x numeric)	numeric	integer quotient of y/x	div(9,4)	2
exp(dp or numeric)	(same as input)	exponential	exp(1.0)	2.71828182845905
floor(dp or numeric)	(same as input)	largest integer not greater than argument	floor(-42.8)	-43
ln(dp or numeric)	(same as input)	natural logarithm	ln(2.0)	0.693147180559945
log(dp or numeric)	(same as input)	base 10 logarithm	log(100.0)	2
log(b numeric, x numeric)	numeric	logarithm to base b	log(2.0, 64.0)	6.0000000000
mod(y, x)	(same as argument types)	remainder of y/x	mod(9,4)	1

Fungsi Matematika (2)

pi()	dp	"π" constant	pi()	3.14159265358979
power(a dp, b dp)	dp	a raised to the power of b	power(9.0, 3.0)	729
power(a numeric, b numeric)	numeric	a raised to the power of b	power(9.0, 3.0)	729
radians(dp)	dp	degrees to radians	radians(45.0)	0.785398163397448
round(dp or numeric)	(same as input)	round to nearest integer	round(42.4)	42
round(v numeric, s int)	numeric	round to s decimal places	round(42.4382, 2)	42.44
sign(dp or numeric)	(same as input)	sign of the argument (-1, 0, +1)	sign(-8.4)	-1
sqrt(dp or numeric)	(same as input)	square root	sqrt(2.0)	1.4142135623731
trunc(dp or numeric)	(same as input)	truncate toward zero	trunc(42.8)	42
trunc(v numeric, s int)	numeric	truncate to s decimal places	trunc(42.4382, 2)	42.43
width_bucket(op numeric, b1 numeric, b2 numeric, count int)	int	return the bucket to which operand would be assigned in an equidepth histogram with count buckets, in the range b1 to b2	width_bucket(5.35, 0.024, 10.06, 5)	3
width_bucket(op dp, b1 dp, b2 dp, count int)	int	return the bucket to which operand would be assigned in an equidepth histogram with count buckets, in the range b1 to b2	width_bucket(5.35, 0.024, 10.06, 5)	3

Conditional : IF .. THEN

39.6.2.1. IF-THEN

```
IF boolean-expression THEN
    statements
END IF;
```

```
IF v_user_id <> 0 THEN
    UPDATE users SET email = v_email WHERE user_id = v_user_id;
END IF;
```

Conditional : IF .. THEN .. ELSE

39.6.2.2. IF-THEN-ELSE

```
IF boolean-expression THEN
    statements
ELSE
    statements
END IF;
```

```
IF v_count > 0 THEN
    INSERT INTO users_count (count) VALUES (v_count);
    RETURN 't';
ELSE
    RETURN 'f';
END IF;
```

Conditional : IF .. THEN .. ELSIF

39.6.2.3. IF-THEN-ELSIF

```
IF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
    ...]]
[ ELSE
    statements ]
END IF;
```

```
IF number = 0 THEN
    result := 'zero';
ELSIF number > 0 THEN
    result := 'positive';
ELSIF number < 0 THEN
    result := 'negative';
ELSE
    -- hmm, the only other possibility is that number is null
    result := 'NULL';
END IF;
```

Conditional : Simple Case

39.6.2.4. Simple CASE

```
CASE search-expression
    WHEN expression [, expression [ ... ]] THEN
      statements
  [ WHEN expression [, expression [ ... ]] THEN
      statements
    ... ]
  [ ELSE
      statements ]
END CASE;
```

```
CASE x
    WHEN 1, 2 THEN
        msg := 'one or two';
    ELSE
        msg := 'other value than one or two';
END CASE;
```

Conditional : Searched Case

39.6.2.5. Searched CASE

```
CASE
    WHEN boolean-expression THEN
      statements
  [ WHEN boolean-expression THEN
      statements
    ... ]
  [ ELSE
      statements ]
END CASE;
```

```
CASE
    WHEN x BETWEEN 0 AND 10 THEN
        msg := 'value is between zero and ten';
    WHEN x BETWEEN 11 AND 20 THEN
        msg := 'value is between eleven and twenty';
END CASE;
```

Studi kasus:

- Buat tabel nilai_siswa:

```
dblatihan=# \d nilai_ujian;
                                  Table "public.nilai_ujian"
   Column    |         Type          |                          Modifiers
-------------+-----------------------+-------------------------------------------------------------
 id          | integer               | not null default nextval('nilai_ujian_id_seq'::regclass)
 kodemk      | character varying(10) |
 nim         | character varying(10) |
 total_nilai | double precision      |
 grade       | character(2)          |
Indexes:
    "nilai_ujian_pkey" PRIMARY KEY, btree (id)
```

```
dblatihan=# SELECT * FROM nilai_ujian;
 id | kodemk |  nim  | total_nilai | grade
----+--------+-------+-------------+-------
  1 | NF0001 | 16001 |          80 | B+
(1 row)
```

Soal:

1. Buat Fungsi : kelulusan , dengan ketentuan >65 dinyatakan ‘LULUS’ dan selainnya ‘TIDAK LULUS’
2. Buat Fungsi : grade , dengan ketentuan :

Nilai Angka	Range Nilai	Nilai Mutu
A	85.01 – 100.00	4.00
A-	80.01 – 85.00	3.70
B+	75.01 – 80	3.30
B	70.01 – 74	3.00
B-	65.01-70	2.70
C+	60.00 – 65	2.30
C	55.01 - 60	2.00

Tugas Baca:

- Referensi:
  - http://www.postgresqltutorial.com/plpgsql-cursor/