Veritabanı uygulama dersi - 7: PL/pgSQL Fonksiyon Tanımı

- Standart postgreSQL'de dört farklı prosedürel dil vardır:
- 1) PL/pgSQL 2) PL/TcL 3)PL/Perl 4) PL/Phyton

- PL/pgSQL'in avantajları:
  - Fonksiyon ve trigger'lar yaratabiliriz;
  - Döngüsel ve koşula bağlı işlem adımları daha kolay yapılabilir;
  - Karmaşık sorgulamalar ve hesaplamalar yapılabilir;
  - Kullanıcının kendi amacına yönelik fonksiyon yazabilmesi sağlanır.
  - PL/pgSQL, SQL'in tüm veri tipi, operatör ve hazır fonksiyonlarını tanır ve kullanabilir.

- Bir PL/pgSQL fonksiyonu sonucunda tek bir değer dönmek zorunda değildir. Birden fazla dönüş olacaksa **output** parametre tanımı kullanılmalıdır. Bunun dışında fonk'lar, basit tipte bir veri dönebilir; record tipinde composit bir data dönebilir; ya da bir sonuç tablosunun pointer'ı gibi tek bir instance dönebilir. Hatta bazen hiç değer dönmeyebilir.

- Hiç bir değer dönmüyor ise sadece "return" ya da "return void" denilebilir fonksiyonun sonunda.

Standart postgreSQL'de prosedürel bir dili kullanmadan önce bu dili oluşturmalıyız:

"CREATE LANGUAGE plpgsql "

Aşağıda fonksiyon oluşturmak için gerekli olan PL/pgSQL syntax'ı verilmiştir. (Köşeli parantezler [ ], opsiyonel durumları göstermektedir.)

CREATE FUNCTION *fonksiyon_adı (parametre1 tipi, parametre2 tipi,..., [out] parametreN tipi)*
[RETURNS *çıktının veri tipi]* **AS [$$] [']**
DECLARE
tanımlamalar;
BEGIN
komutlar;
..
[RETURN] *[çıktı değeri;]*
..
EXCEPTION
kural dışı durumlar;
END;
[$$] ['] LANGUAGE plpgsql;

Fonksiyonu çağırmak için:
SELECT *fonksiyon_adı (parametre değerleri);*
Fonksiyonu düşürmek için:
DROP FUNCTION *fonksiyon_adı (parametre1_tipi, parametre2_tipi, ..., parametreN_tipi);*

RETURN - Fonksiyondan dönme - çeşitleri

Bir PL/pgSQL fonksiyonu bittikten sonra fonksiyondan çıkabilmek için "RETURN expression" kullanılır.

Ancak fonksiyon ile birden fazla değer değiştirilecek veya **output** tipinde parametre tanımlamaları yapılmış ise return kullanılmaz ya da sadece return denir; yanına expression yazılmaz.

Bunun dışında fonksiyon tanımlanırken return void denildiyse, yine return kullanılmak zorunda değildir. Fonksiyondan erken çıkabilmek için sadece return denebilir yanına expression yazılmaz (yukarıda tanımlandığı gibi)

Output parametresi kullanan fonksiyonlar ve hiç bir şey döndürmeyen (void) fonksiyonlar dışında, her fonksiyonun return parametresi olmak zorundadır. Değilse run time hatası alınır.

Begin- End Blokları içinde Kullanılabilecek Statement ve Loop'ların Kod Tanımları

<table>
<tr>
<td>Blok yapısı:
[DECLARE]
BEGIN
….
[DECLARE]
BEGIN
…
END;

[EXCEPTION]
….
END;</td>
<td>Declare
DECLARE
variable1 type := initial_value1;

Examples of variable declarations:
user_id integer;
quantity numeric(5);
url varchar;
my_var tablename.columnname%TYPE;</td>
<td>If Statement

IF condition THEN
Statements;
[ELSIF condition THEN
Statements;]
[ELSE
statements;
]
END IF;</td>
<td>Case Statement
CASE selector
WHEN expression1 THEN satatement(s)1;
WHEN expression2 THEN satatement(s)2;
….
WHEN expressionN THEN satatement(s)N;

[ELSE statement(s)N+1]
END;</td>
</tr>
<tr>
<td colspan="2">LOOP
Statement1;
….
EXIT [WHEN condition];
END LOOP;</td>
<td>WHILE condition LOOP
Statement1;
Statement2;
…
END LOOP;</td>
<td>FOR counter IN [REVERSE] lower_limit..upper_limit LOOP
Statement1;
Statement2;
…
END LOOP;</td>
</tr>
<tr>
<td colspan="4">SELECT column1, column2, … column INTO var1, var2,… varN FROM table [WHERE]</td>
</tr>
</table>

Örnekler

1. Girdi olarak verilen 2 sayının toplamını bulan fonksiyonu yazınız ve (22,63) parametreleri için çalıştırınız.

2. Adı verilen bir departmandaki çalışanların ortalama maaşını bulan bir fonksiyon yazınız.

3. Departman tablosundaki minimum ve maksimum departman numarasını bulup min_deptno ve max_deptno değişkenlerine atan fonksiyonu yazınız.

4. 6 no'lu departmanda çalışanların sayısını bulun, çalışan sayısı 10'dan azsa departmandaki tüm çalışanların maaşına %5 zam yapın.

Cevaplar

1. CREATE OR REPLACE FUNCTION ornek1 (num1 numeric, num2 numeric) RETURNS numeric AS '
DECLARE
toplam numeric;
BEGIN
toplam :=num1+num2;
return toplam;
END;
' LANGUAGE 'plpgsql';

NOT: AS'den sonra ve LANGUAGE'ten önce tırnak (') yerine $$ işaretleri de gelebilirdi.
Çağırılması: select ornek1(22,63);

Return olmadan - Out ile - Çözüm:
CREATE or replace FUNCTION ornek1(num1 numeric, num2 numeric, out num3 numeric) AS '

```
BEGIN
        num3 :=num1+num2;
END;
' LANGUAGE  'plpgsql';
```

Çağırılması yukarıdakinin aynısıdır. **Düşürme:** DROP FUNCTION ornek1 (numeric, numeric)

2. CREATE OR REPLACE FUNCTION ornek2 (depname department.dname%type) RETURNS real AS '
DECLARE
 maas numeric;
 BEGIN
        SELECT AVG(salary) INTO maas FROM employee, department WHERE dno = dnumber AND dname = depname;
        RETURN maas;
 END;
' LANGUAGE 'plpgsql';

Çağırılması: select ornek2('Sales'). **Düşürme:** DROP FUNCTION ornek2 (department.dname%type)

3. CREATE OR REPLACE FUNCTION ornek3 (out min_deptno department.dnumber%type, out max_deptno department.dnumber%type) AS '
BEGIN
        Select min(dnumber), max(dnumber) INTO min_deptno, max_deptno from department;
END;
' LANGUAGE 'plpgsql';
Çağırılması: select ornek3(). **Düşürme:** DROP FUNCTION ornek3 ()

4. CREATE OR REPLACE FUNCTION ornek4 () returns void AS '
DECLARE
num_worker numeric(3) := 0;
BEGIN
        Select count(*) into num_worker from employee where dno=6;
        IF (num_worker < 10) THEN
                Update employee SET salary=salary*1.05 where dno=6;
        END IF;
END;
' LANGUAGE 'plpgsql';
Çağırılması: select ornek4(). **Düşürme:** DROP FUNCTION ornek4 ()