Veri Tabanı Dersi 6. Laboratuvarı

Grup 7,8

Arş. Gör. Furkan Çakmak

Laboratuvar Programı

VT 20161
Lab 6

- Hafta 1 - SQL'e giriş; DDL ve DML komutlarına giriş
- Hafta 2 - Postgresql ortamının tanıtımı, Company-db'nin tanıtımı ve Sorgulama örnekleri
- Hafta 3 - Tablolarda Kısıt, View ve Sequence İşlemleri; Union, Intersect, Except İşlemleri
- Hafta 4 - Tablolarda Gruplama ve Sıralama Fonksiyonları
- Hafta 5 - JDBC ile Veri Tabanına Bağlanıp Sorgu Yapma Uygulamaları
- Hafta 6 - PL/pgSQL Fonksiyon Tanımı
- Hafta 7 - PL/pgSQL Alias, Record/Cursor ve Trigger Tanımları

PostgreSQL Fonksiyonel Dilleri ve PL/pgSQL

VT 20161
Lab 6

PostgreSQL'de 4 farklı fonsiyonel (prosedürel) dil var.

1. PL/pgSQL
2. PL/TcL (C programlama dili)
3. PL/Perl (Perl Programlama Dili)
4. PL/Phyton (Phyton Programlama Dili)
5. «SELECT * FROM pg_pltemplate;»

PL/pgSQL Avantajları:

1. Fonksiyonlar ve trigger'lar oluşturulabilir.
2. Döngüsel ve koşula bağlı işlem adımları daha kolay yapılabilir. (while, for, if)
3. Karmaşık sorgular ve hesaplamalar yapılabilir.
4. Kullanıcının kendi amacına yönelik fonksiyon yazması sağlanabilir.
5. PL/pgSQL, SQL'in tüm veri tipi, operatör ve hazır fonksiyonlarını tanır ve kullanabilir.

PL/pgSQL Fonksiyon Dönüş Tipleri

VT 20161
Lab 6

1. PL/pgSQL tek bir değer döndürmek zorunda değildir.
2. Birden fazla dönüş yapılacaksa «output» anahtar sözlüğü kullanılır.
3. PL/pgSQL fonksiyonları basit tipte veri döndürebileceği gibi bileşik (composit) bir veri de döndürebilirler.
4. Ya da bir sonuç kümesinin (tablosunun) adresini gösteren bir işaretçi (pointer) döndürebilir.
5. Bütün fonksiyonlar da değer döndürmek zorunda değildir (return, return void).

PL/pgSQL Sintaksı

VT 20161
Lab 6

PostgreSQL'de fonksiyonel bir dilin kullanılabilmesi için «**CREATE LANGUAGE plpgsql**» komutu kullanılarak bu dil öncelikle oluşturulmalıdır.

```
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_adi(parametreler);
```

Düşürmek için;

```
DROP FUNCTION fonksiyon_adi(parametreler);
```

PL/pgSQL RETURN ve Dönüş Çeşitleri

VT 20161
Lab 6

- PL/pgSQL fonksiyonundan çıkış için «RETURN 'ifade'» anahtar sözcüğü kullanılır.
- Eğer parametre döndürülmeyecekse sadece «RETURN» yazılır. Veya birden çok parametre döndürülecekse «OUTPUT» anahtar sözcüğü kullanılabilir.
- «OUTPUT» veya «RETURN VOID» olmayan fonksiyonlarda [illegible] ifadesi olmak zorundadır. Yoksa çalışma sırasında hata ile karşılaşılır.

PL/pgSQL Statement ve Loop Sintaksı

VT 20161
Lab 6

Değişken Tanımı:

- user_id integer;
- quantity numeric(5);
- url varchar;
- my_var tablename.columnname %TYPE;

«IF» Koşulu Tanımı:

```
IF koşul THEN yapılacaklar;
[ELSEIF koşul THEN yapılacaklar;]
[ELSE yapılacaklar;]
END IF;
```

«CASE» Tanımı:

```
CASE secici
WHEN secici_kosulu1 THEN yapilacaklar1;
WHEN secici_kosulu2 THEN yapilacaklar2;
…
WHEN secici_kosuluN THEN yapilacaklarN;
[ELSE secici_kosulu(N+1)]
END;
```

«LOOP» Tanımı:

```
LOOP
        yapilacaklar…
EXIT [WHEN dongu_kosulu];
END LOOP;
```

«FOR» Tanımı:

```
FOR sayaç IN [REVERSE]
alt_limit…ust_limit LOOP
        yapilacaklar…
END LOOP;
```

«WHILE» Tanımı:

```
WHILE dongu_kosulu LOOP
        yapilacaklar…
END LOOP;
```

PL/pgSQL Örnekleri

VT 20161
Lab 6

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 min_deptno ve max_deptno değişkenlerine atan fonksiyonu yazınız.

4. Numarası verilen departmandaki çalışanların sayısını bulun, çalışan sayısı 10'dan azsa departmandaki tüm çalışanların maaşına zam yapın.

5. İsmi verilen bir departmanda çalışanların ortalama maaşı, verilen bir değerden düşük ve o departmandaki kadın çalışanların maaşlarının toplamı verilen bir limitin üstündeyse, o departmanda 1'den fazla projede çalışanların maaşlarına yine verilen bir oranda zam yapan fonksiyonu yazınız.

```
SELECT kosullu_zam_yap('Research', 50000, 20000, 5);
```

Sabırla Dinlediğiniz İçin Teşekkürler

VT 20161
Lab 6

PL/pgSQL Örnekleri Cevap 1

VT 20161
Lab 6

```
-- RETURN ile çözüm
CREATE FUNCTION toplama(sayi1 integer, sayi2 integer) RETURNS integer AS '
DECLARE
        toplam integer;
BEGİN
        toplam := sayi1 + sayi2;
        return toplam;
END;
' LANGUAGE 'plpgsql';

/*Çağırılması : */      SELECT toplama1(22,63);
/*Düşürülmesi: */       DROP FUNCTION toplama1(integer,integer);
```

```
-- OUT ile çözüm
CREATE FUNCTION toplama(sayi1 integer, sayi2 integer, OUT toplam integer) AS '
BEGİN
        toplam := sayi1 + sayi2;
END;
' LANGUAGE 'plpgsql';

SELECT toplama(22,63);
/*Çağırılması : */      SELECT toplama2(22,63);
/*Düşürülmesi: */       DROP FUNCTION toplama2(integer,integer,out
integer); /*veya*/ DROP FUNCTION toplama1(integer,integer);
```

PL/pgSQL Örnekleri Cevap 2

VT 20161
Lab 6

```
CREATE FUNCTION dep_ort_maas(depname department.dname%TYPE, out ort_maas real) AS '
BEGIN
        SELECT AVG(salary) INTO ort_maas FROM employee e, department d WHERE e.dno = d.dnumber AND dname = depname;
END;
' LANGUAGE 'plpgsql';

/*Çağırılması : */        SELECT dep_ort_maas('Sales');
/*Düşürülmesi: */        DROP FUNCTION dep_ort_maas(department.dname%TYPE, out real); /*veya*/ DROP FUNCTION
dep_ort_maas(department.dname%TYPE);
```

PL/pgSQL Örnekleri Cevap 3

VT 20161
Lab 6

```
CREATE FUNCTION dep_no_bul(out min_deptno numeric, out max_deptno numeric) AS '
BEGIN
        SELECT MIN(dnumber),MAX(dnumber) INTO min_deptno, max_deptno FROM department;
END;
' LANGUAGE 'plpgsql';

/*Çağırılması : */        SELECT dep_no_bul();
/*Düşürülmesi: */        DROP FUNCTION dep_no_bul(out numeric, out numeric); /*veya*/ DROP FUNCTION dep_no_bul();
```

PL/pgSQL Örnekleri Cevap 4

VT 20161
Lab 6

```
CREATE FUNCTION dep6_emp(dnum numeric) RETURNS VOID AS '
DECLARE
        emp_sayac numeric;
BEGIN
        SELECT count(*) INTO emp_sayac FROM employee e WHERE e.dno = dnum;
        IF(emp_sayac < 10) THEN
                UPDATE employee e SET salary = salary*1.05 WHERE e.dno = dnum;
        END IF;
END;
' LANGUAGE 'plpgsql';

/*Çağırılması : */      SELECT dep6_emp(6);
/*Düşürülmesi: */       DROP FUNCTION dep6_emp(numeric);
```

PL/pgSQL Örnekleri Cevap 5

VT 20161
Lab 6

```
CREATE OR REPLACE FUNCTION kosullu_zam_yap(bolum_ismi department.dname%TYPE, ort_maas real, f_top_maas employee.salary%TYPE, zam_orani
integer) RETURNS VOID AS '
DECLARE
        ger_ort_maas real;
        kadin_maaslari integer;
        bolum_no department.dnumber%TYPE;
BEGIN
        SELECT dnumber INTO bolum_no FROM department WHERE dname = bolum_ismi;
        SELECT AVG(salary) INTO ger_ort_maas FROM employee WHERE dno = bolum_no;
        SELECT SUM(salary) INTO kadin_maaslari FROM employee WHERE dno = bolum_no AND sex = ''F'';
        IF ger_ort_maas < ort_maas AND kadin_maaslari > f_top_maas THEN
                UPDATE employee SET salary = salary*zam_orani/100 + salary WHERE ssn IN (SELECT essn FROM employee, works_on WHERE
                        ssn = essn AND dno = bolum_no GROUP BY essn HAVING COUNT(*) > 1);
        END IF;
END;
'LANGUAGE plpgsql;
```