Veri Tabanı Dersi 7. Laboratuvarı

Grup 2 (UK)

Arş. Gör. Furkan Çakmak

Laboratuvar Programı

VT 20171
Lab 7

- 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 – Quiz 1
- Hafta 5 – Tablolarda Gruplama ve Sıralama Fonksiyonları
- Hafta 6 - JDBC ile Veri Tabanına Bağlanıp Sorgu Yapma Uygulamaları
- Hafta 7 - PL/pgSQL Fonksiyon Tanımı
- Hafta 8 – Quiz 2
- Hafta 9 - PL/pgSQL Alias, Record/Cursor ve Trigger Tanımları
- Hafta 10 - Xquery Yapısı ve Örnekleri
- Hafta 11 – Quiz 3

PL/pgSQL Record ve Cursor Tanımı

VT 20171
Lab 7

1. Geçtiğimiz hafta bahsettiğimiz gibi, PL/pgSQL fonksiyonlarını sadece tek bir değer döndürmek zorunda değildi. Karmaşık sonuçları veya bir tabloyu da döndürebileceğimizden bahsetmiştik.
2. Bu tarz composit veri tiplerini döndürmek için RECORD tanımları kullanılmaktadır.
3. Eğer bir tablo döndürmek istiyorsak CURSOR tanımlayarak sonucu bunun üzerinden döndürebiliriz.

RECORD Tanımı:

CREATE TYPE my_record_type AS (field1 type1, field2 type2, …);

Ör: CREATE TYPE emp_kisa_bilgi AS (fname varchar(15), salary integer);

CURSOR Tanımı:

cursor_name [[NO] SCROLL] CURSOR [(arguments)] FOR sql_query;

Ör: my_cur CURSOR FOR select * from employee;

CURSOR kullanımı:

OPEN my_cursor;

FETCH [direction { FROM | IN }] cursor INTO target;

Target: record ya da variable1, variable2,.. şeklinde olabilir.

MOVE [direction { FROM | IN }] cursor;

CLOSE my_cursor;

PL/pgSQL RAISE Tanımı

VT 20171
Lab 7

RAISE Tanımı: (Ekrana mesaj yazdırmak için kullanılır.)

RAISE [level] 'format' [, expression [, ...]] [USING option = expression [, ...]];

RAISE [level] condition_name [USING option = expression [, ...]];

RAISE [level] SQLSTATE 'sqlstate' [USING option = expression [, ...]];

RAISE [level] USING option = expression [, ...];

RAISE ;

Ör: RAISE NOTICE 'Salary here is %', sal_variable;

"Level" seçeneği, ne tür mesaj yazdırmak istediğimizi söyler:

- **DEBUG,**
- **LOG,**
- **INFO,**
- **NOTICE,**
- **WARNING,**
- **EXCEPTION**
- **Default'u EXCEPTION'dır.**

PL/pgSQL Record ve Cursor Örnekleri 1

VT 20171
Lab 7

Örnek 1: SSN'i parametre olarak verilen çalışanın ismini, çalıştığı departmanın ismini ve maaşını ekrana yazdıran PL/pgSQL bloğunu yazın. Bir ssn vererek fonksiyonu çağırınız.

```
CREATE TYPE record1 AS (isim varchar(15), bolum varchar(25), maas integer);
CREATE OR REPLACE FUNCTION calisan_bilgiler(emp_ssn employee.ssn%TYPE) RETURNS record1 AS '
DECLARE
        emp_record record1;
BEGIN
        SELECT fname, dname, salary INTO emp_record FROM employee e, department d WHERE e.dno = d.dnumber AND
e.ssn = emp_ssn;
        RAISE NOTICE "Calisan ismi: %, bolum ismi: %, maas: %", emp_record.isim, emp_record.bolum, emp_record.maas;
        RETURN emp_record;
END;
' LANGUAGE 'plpgsql';
/*Çağırılması: */     SELECT calisan_bilgiler('123456789');
/*Düşürülmesi: */     DROP FUNCTION calisan_bilgiler(employee.ssn%type); DROP TYPE record1;
```

PL/pgSQL Record ve Cursor Örnekleri 2

VT 20171
Lab 7

Örnek 2: Numarası verilen bir departmandaki çalışanların isimlerini bulan bir fonksiyon yazınız. Bir departman numarası vererek fonksiyonu çağırınız.

```
CREATE OR REPLACE FUNCTION bolum_calisanlari(dnum numeric) RETURNS void AS '
DECLARE
        emp_cursor CURSOR FOR SELECT fname, lname FROM employee WHERE dno = dnum;
BEGIN
        FOR emp_record IN emp_cursor LOOP
                RAISE INFO "Calisan ismi: % %",emp_record.fname, emp_record.lname;
        END LOOP;
END;
' LANGUAGE 'plpgsql';
/*Çağırılması: */     SELECT bolum_calisanlari(6);
/*Düşürülmesi: */     DROP FUNCTION bolum_calisanlari(numeric);
```

PL/pgSQL Record ve Cursor Örnekleri 3

VT 20171
Lab 7

Örnek 3: Numarası verilen bir projede çalışanların maaşları verilen bir değere tam bölünebiliyorsa, o kişilerin ad, soyad ve maaş bilgilerini HAVING fonksiyonu kullanmadan listeleyen fonksiyonu yazınız.

```
CREATE TYPE calisan AS (isim varchar(15), soyisim varchar(15), maas integer);
CREATE OR REPLACE FUNCTION calisan_listele(pnum project.pnumber%TYPE, bolen integer) RETURNS calisan[] AS '
DECLARE
        emp_cursor CURSOR FOR SELECT fname, lname, salary FROM employee, works_on WHERE ssn = essn AND pno = pnum;
        cal calisan[];
        i integer;
BEGIN
        i := 1;
        FOR emp_record IN emp_cursor LOOP
                IF emp_record.salary % bolen = 0 THEN
                        cal[i] = emp_record;
                        i := i + 1;
                END IF;
        END LOOP;
        RETURN cal;
END;
' LANGUAGE 'plpgsql';
/*Çağırılması: */       SELECT calisan_listele('61',9);
/*Düşürülmesi: */       DROP FUNCTION calisan_listele(project.pnumber%TYPE, integer);
```

PL/pgSQL Record ve Cursor Örnekleri 4

VT 20171
Lab 7

Örnek 4: Departman numarası verilen bir departmandaki çalışanların toplam maaşını (SUM() fonksiyonundan yararlanmadan) bulan bir fonksiyon yazınız. (SELECT sum(salary) FROM employee WHERE dno = X;)

```
CREATE OR REPLACE FUNCTION dep_sum_salary(dnum numeric, OUT sum_sal numeric) AS '
DECLARE
        emp_cursor CURSOR FOR SELECT salary FROM employee WHERE dno = dnum;
BEGIN
        sum_sal := 0;
        FOR emp_record IN emp_cursor LOOP
                sum_sal := sum_sal + emp_record.salary;
        END LOOP;
END;
' LANGUAGE 'plpgsql';
/*Çağırılması: */       SELECT dep_sum_salary(6);
/*Düşürülmesi: */       DROP FUNCTION dep_sum_salary(numeric);
```

PL/pgSQL Trigger Tanımı

VT 20171
Lab 7

1. Fonksiyonlar gibi veri tabanına kaydedilirler.
2. VTYS tarafından trigger'ın şartları oluştuğunda otomatik olarak çağırılırlar.
3. DML komutları trigger'ı başlatır. (INSERT, UPDATE, DELETE, vb.)

CREATE TRIGGER *trigger_isim* { **BEFORE** | **AFTER** }
{ ***event1*** [OR ***event2*** OR ...] } **ON** *tablo_adı*
[FOR [EACH] { ROW | STATEMENT }] [WHEN (*condition*)]
EXECUTE PROCEDURE *trigger_fonk_adi(arguments*);

```
CREATE OR REPLACE FUNCTION trig_fonk()
RETURNS TRIGGER AS '
BEGIN
Statements;
[ RETURN [NULL | OLD | NEW]; ]
END;
' LANGUAGE 'plpgsql';
```

PL/pgSQL Trigger Tanımı (Devam)

VT 20171
Lab 7

Part	Description	Possible Values
Trigger timing	Trigger'ın harekete geçtiği an	Before / After
Trigger event	Trigger'ı tetikleyen DML	Insert / Update / Delete
Trigger type	Trigger body'nin çalışma sayısı	Statement / Row

Trigger tipi, trigger fonksiyonunun, bir SQL sorgusu için **sadece bir kez** mi, yoksa trigger olayından etkilenen **her bir satır** için mi çalışacağını belirler. Varsayılanı "FOR EACH **STATEMENT**"tır.

NEW: Tetikleyici prosedürün/fonksiyonun body bloğunda kullanılır. Row-level tetikleyiciler için insert/update olaylarında yeni eklenen satırın değerini tutan record yapısındaki değişkendir. Statement-level tetikleyicilerde ve Delete işlemlerinde NEW değişkeni NULL'dır.

OLD: Tetikleyici prosedürün/fonksiyonun body bloğunda kullanılır. Row-level tetikleyiciler için update/delete olaylarında, değişen/silinen eski satırın değerini tutan record yapısındaki değişkendir. Statement-level tetikleyicilerde ve Insert işlemlerinde OLD değişkeni NULL'dır.

Trigger düşürülmesi: **DROP TRIGGER trigger_fonk_adi ON tablo_adı [CASCADE | RESTRICT]**

CASCADE: Tetikleyiciye bağlı olan nesneleri de otomatik olarak düşürür.

RESTRICT: Eğer tetikleyiciye bağlı nesneler varsa tetikleyici düşürülmez. Varsayılanı budur.

PL/pgSQL Trigger Örnekleri 1

VT 20171
Lab 7

Örnek 1: Sadece tatil günleri dışında ve mesai saatleri içinde employee tablosuna insert yapılmasına izin veren trigger'ı yazınız.

```
CREATE OR REPLACE FUNCTION trigger_fonkiyonu_1() RETURNS TRIGGER AS '
BEGIN
        IF (to_char(NOW(), "DY") IN ("SAT", "SUN") OR to_char(NOW(), "HH24") NOT BETWEEN "08" AND "18") THEN
                RAISE EXCEPTION "Sadece mesai gunlerinde ve mesai saatlerinde insert yapabilirsiniz.";
                RETURN NULL;
        ELSE
                RETURN NEW;
        END IF;
END;
' LANGUAGE 'plpgsql';
CREATE TRIGGER t_orn1 BEFORE INSERT ON employee FOR EACH ROW EXECUTE PROCEDURE trigger_fonkiyonu_1();
/*Düşürülmesi: Önce:*/ DROP TRIGGER t_orn1 on employee; /*Sonra:*/ DROP FUNCTION trigger_fonkiyonu_1();
/*Tetiklenmesi:*/INSERT INTO employee VALUES('Vlademir', 'S', 'Putin', '666666666', '1952-10-07', '8975 Suriye', 'M', '125000', '333445555', '5');
```

PL/pgSQL Trigger Örnekleri 2

VT 20171
Lab 7

Örnek 2: Proje tablosunda pnumber kolonu değişince works_on tablosunda da pno'nun aynı şekilde değişmesini sağlayan trigger'ı yazınız. (Öncelikle works_on tablosundan pno'nun yabancı anahtar olma kısıtını kaldırınız.)

```
CREATE OR REPLACE FUNCTION trigger_fonkiyonu_2() RETURNS TRIGGER AS '
BEGIN
        UPDATE works_on SET pno = NEW.pnumber WHERE pno = OLD.pnumber;
        RETURN NEW;
END;
'LANGUAGE plpgsql;

CREATE TRIGGER t_orn2 AFTER UPDATE ON project FOR EACH ROW EXECUTE PROCEDURE trigger_fonkiyonu_2();
/*Düşürülmesi: Önce:*/ DROP TRIGGER t_orn2 on department; /*Sonra:*/ DROP FUNCTION trigger_fonkiyonu_2();
/*Tetiklenmesi:*/ UPDATE project SET pnumber = 999 WHERE pnumber = 61;
```

PL/pgSQL Trigger Örnekleri 3

VT 20171
Lab 7

Örnek 3: Maaş inişine ve %10'dan fazla maaş artışına izin vermeyen trigger'ı yazınız.

```
CREATE OR REPLACE FUNCTION trigger_fonkiyonu_3() RETURNS TRIGGER AS '
BEGIN
        IF (OLD.salary > NEW.salary OR NEW.salary > 1.1*OLD.salary) THEN
                RAISE EXCEPTION "Maasi dusuremezsiniz ve %%10"dan fazla zam yapamazsınız.";
                RETURN OLD;
        ELSE
                RETURN NEW;
        END IF;
END;
' LANGUAGE 'plpgsql';
CREATE TRIGGER t_orn3 BEFORE UPDATE ON employee FOR EACH ROW EXECUTE PROCEDURE trigger_fonkiyonu_3();
/*Düşürülmesi: Önce:*/ DROP TRIGGER t_orn3 on employee; /*Sonra:*/ DROP FUNCTION trigger_fonkiyonu_3();
/*Tetiklenmesi*/ UPDATE employee SET salary = salary*1.12;
```

PL/pgSQL Trigger Örnekleri 4

VT 20171
Lab 7

Örnek 4: Departman tablonuza salary ile aynı tipte total_salary kolonu ekleyin. Employee tablosunda bir işçinin maaşında maaş değişikliği olduğunda departman tablonuzdaki total_salary kolonunda da gerekli güncellemeyi yapacak trigger'ı yazınız.

```
ALTER TABLE department ADD COLUMN total_salary; UPDATE department SET total_salary = (SELECT SUM(salary) FROM employee WHERE dno = dnumber);
```

PL/pgSQL Trigger Örnekleri 4

VT 20171
Lab 7

Örnek 4: Departman tablonuza salary ile aynı tipte total_salary kolonu ekleyin. Employee tablosunda bir işçinin maaşında maaş değişikliği olduğunda departman tablonuzdaki total_salary kolonunda da gerekli güncellemeyi yapacak trigger'ı yazınız.

```
ALTER TABLE department ADD COLUMN total_salary; UPDATE department SET total_salary = (SELECT SUM(salary) FROM employee WHERE dno = dnumber);
CREATE OR REPLACE FUNCTION trigger_fonkiyonu_4() RETURNS TRIGGER AS $$
BEGIN
        IF (TG_OP = 'DELETE') THEN
                UPDATE department SET total_salary = total_salary - OLD.salary WHERE dnumber = OLD.dno;
        ELSEIF (TG_OP = 'UPDATE') THEN
                UPDATE department SET total_salary = total_salary - OLD.salary + NEW.salary WHERE dnumber = OLD.dno;
        ELSE
                UPDATE department SET total_salary = total_salary + NEW.salary WHERE dnumber = NEW.dno;
        END IF;
        RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
CREATE TRIGGER t_orn4 AFTER INSERT OR UPDATE OR DELETE ON employee FOR EACH ROW EXECUTE PROCEDURE trigger_fonkiyonu_4();
 /*Tetiklenmesi 1*/ INSERT INTO employee VALUES('Vlademir', 'S', 'Putin', '666666667', '1952-10-07', '8975 Suriye', 'M', '100000000', '333445555', '1');
/*Tetiklenmesi 2,3*/ UPDATE employee SET salary = salary*1.07 WHERE dno = 1; DELETE FROM employee WHERE ssn = '111111103';
```

Sabırla Dinlediğiniz İçin Teşekkürler

VT 20171
Lab 7