VERİTABANI DERSİ 8. LABORATUVARI

8.Lab Dersi
Trigger Tanımları

TRIGGERS (tetikleyiciler)

1. Fonksiyonlar gibi veri tabanına kaydedilirler.
2. VTYS tarafından **trigger'ın şartları oluştuğunda** otomatik **olarak çağrılırlar.**
3. Tablolar üzerinde değişiklik yapılmak istendiğinde çalışırlar.

INSERT, UPDATE, DELETE

CREATE TRIGGER *trigger_isim*

{ **BEFORE** | **AFTER** } { ***events*** }

ON *tablo_adı*

FOR EACH ROW EXECUTE PROCEDURE ***trigger_fonk_ad****i*();

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

Trigger fonksiyonları:

Parametre almazlar
Trigger döndürürler.

1. Tablo ile trigger fonksiyonu bağlanır:

```
CREATE TRIGGER trig_isim
```

2. Trigger fonksiyonu yazılır.

```
CREATE or REPLACE FUNCTION trig_fonk_isim()
```

INSERT = Sadece NEW kullanılır.

UPDATE = OLD ve NEW kullanılır.

DELETE = Sadece OLD kullanılır.

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.

Örnek – 6

- Sadece tatil günleri dışında ve mesai saatleri içinde employee tablosuna çalışan eklenmesine izin veren trigger'ı yazınız.

```
CREATE  TRIGGER t_ornek6
BEFORE INSERT
ON employee
FOR EACH ROW EXECUTE PROCEDURE trig_fonk_ornek6();
```

```
CREATE FUNCTION trig_fonk_ornek6()
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';
```

Tetiklenmesi:

```
INSERT INTO employee VALUES('Vlademir', 'S', 'Putin', '666666666', '1952-10-07', '8975 Rusya', 'M', '125000', '333445555', '5');
```

Düşürülmesi:

Önce:

```
DROP TRIGGER t_ornek6 on employee;
```

Sonra:

```
DROP FUNCTION trig_fonk_ornek6();
```

Örnek – 7

- Departman tablosunda dnumber kolonu değişince employee tablosunda da dno'nun aynı şekilde değişmesini sağlayan trigger'ı yazınız. (Öncelikle departman tablosundaki yabancı anahtar olma kısıtlarını kaldırınız.)

- **ALTER TABLE** project **DROP CONSTRAINT** project_dnum_fkey;
- **ALTER TABLE** dept_locations **DROP CONSTRAINT** dept_locations_dnumber_fkey;
- **ALTER TABLE** employee **DROP CONSTRAINT** foreign_key_const;

CREATE TRIGGER t_ornek7

AFTER UPDATE

ON department

FOR EACH ROW EXECUTE PROCEDURE trig_fonk_ornek7();

```
CREATE FUNCTION trig_fonk_ornek7()
RETURNS TRIGGER AS $$
BEGIN

    UPDATE      employee
    SET  dno = new.dnumber
    WHERE       dno = old.dnumber;


    RETURN new;
END;
$$ LANGUAGE  'plpgsql';
```

Tetiklenmesi:

```
UPDATE   department
      SET dnumber = 2
      WHERE   dnumber = 5;
```

Örnek – 8

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

```
CREATE  TRIGGER t_ornek8
BEFORE UPDATE
ON employee
FOR EACH ROW EXECUTE PROCEDURE trig_fonk_ornek8();
```

```
CREATE FUNCTION trig_fonk_ornek8()
RETURNS TRIGGER AS $$
BEGIN
    IF(   old.salary > new.salary OR new.salary>1.1*old.salary) THEN
          RAISE EXCEPTION 'Maasi dusuremezsiniz ve %%10dan fazla zam yapamazsınız.';
          RETURN old;
    ELSE
          RETURN new;
    END IF;
END;
$$ LANGUAGE  'plpgsql';
```

Tetiklenmesi:

```
UPDATE employee SET salary = salary*1.12;
```

Düşürülmesi:

Önce:

```
DROP TRIGGER t_ornek8 ON employee;
```

Sonra:

```
DROP FUNCTION trig_fonk_ornek8();
```

Örnek – 9

- Departman tablonuza salary ile aynı tipte `total_salary` kolonu ekleyin. Employee tablosunda maaş değişikliği olduğunda departman tablonuzdaki `total_salary` kolonunda da gerekli güncellemeyi yapacak trigger'ı yazınız.

```
ALTER TABLE department
ADD COLUMN total_salary INTEGER default 0;
```

```
UPDATE department

SET total_salary = (SELECT SUM(salary) FROM employee WHERE dno = dnumber);
```

```
CREATE  TRIGGER t_ornek9
AFTER INSERT or UPDATE or DELETE
ON employee
FOR EACH ROW EXECUTE PROCEDURE trig_fonk_ornek9();
```

```
CREATE FUNCTION trig_fonk_ornek9()
RETURNS TRIGGER AS $$
BEGIN
    IF  (TG_OP = 'DELETE') THEN
        update    department
        set       total_salary=total_salary-old.salary
        where     dnumber=old.dno;
    ELSIF (TG_OP = 'UPDATE') THEN
        update    department
        set       total_salary=total_salary-old.salary+new.salary
        where     dnumber=old.dno;
    ELSE
        update    department
        set       total_salary=total_salary+new.salary
        where     dnumber=new.dno;
    END IF;
    RETURN new;
END;
$$ LANGUAGE  'plpgsql';
```

Tetiklenmesi 1:

INSERT INTO employee **VALUES**('Vlademir', 'S', 'Putin', '666666667', '1952-10-07', '8975 Rusya', 'M', '100000000', '333445555', '1');

Tetiklenmesi 2:

UPDATE employee **SET** salary = salary*1.07 **WHERE** dno = 1;

Tetiklenmesi 3:

DELETE FROM employee **WHERE** ssn = '111111103';

TEŞEKKÜRLER..