HOBİ PAZARI

Ders: Veritabanı Yönetimi

Dersin eğitmeni: Mustafa Utku Kalay

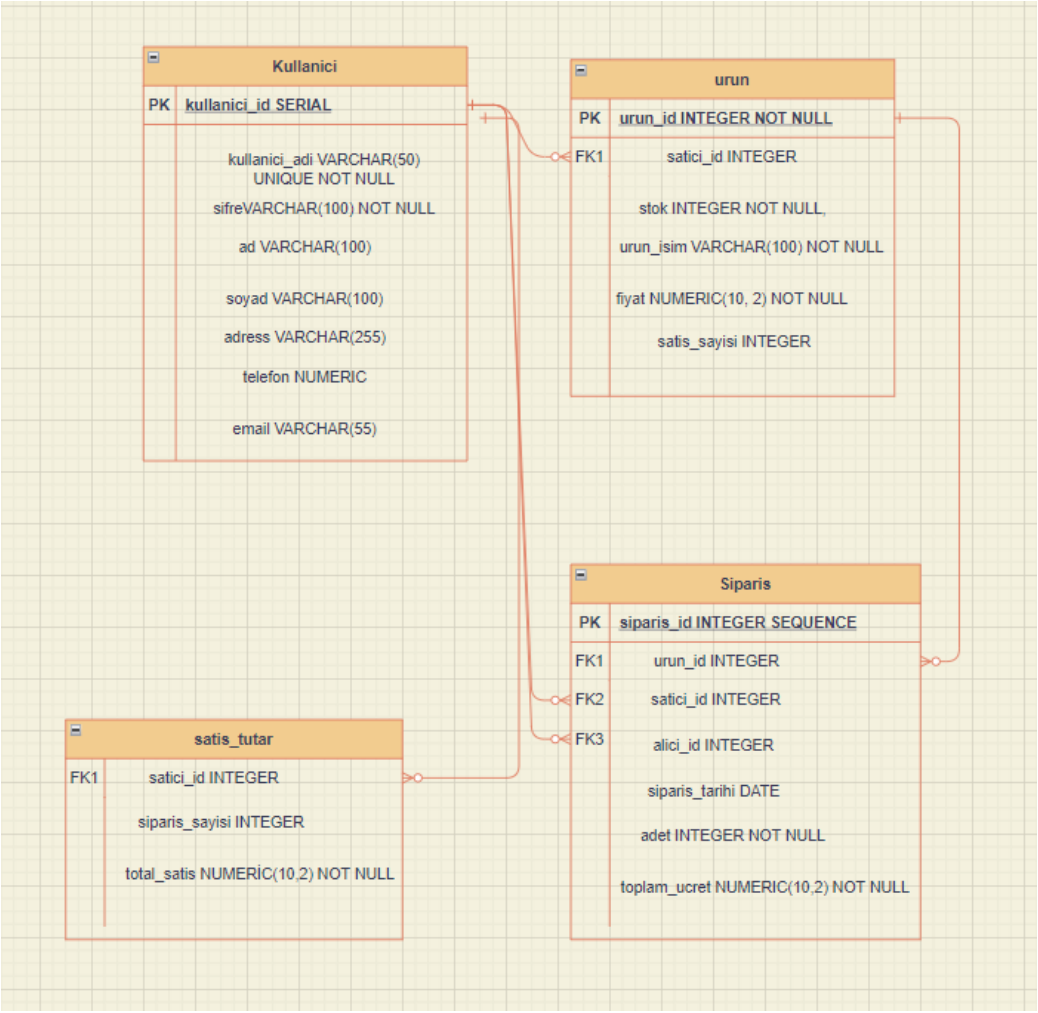
PROJE GRUBU: G11

PROJE KONUSU: Hobi Pazarı & El Emeği Ürünlerin Sergi Satış Sistemi

PROJE EKİBİ

- Emir Öztürk – 20011005
- Eray Gökçe – 20011079
- Emir Can – 20011090
- Eda Altun – 20011092

ER DİYAGRAM



TABLOLAR

1. Kullanıcı tablosu:

	kullanici_id [PK] integer	kullanici_adi character varying (50)	sifre character varying (50)	ad character varying (100)	soyad character varying (100)	adres character varying (255)	telefon numeric	email character varying (55)
1	20011000	eda	eda123	Eda	Altun	123 Main St	5551234567	eda.altun@example.com
2	20011001	emir	emir123	Emir	Can	456 Oak St	5552345678	emir.can@example.com
3	20011002	emirOzturk	emirOzturk123	Emir	Ozturk	789 Pine St	5559876543	emir.ozturk@example.com
4	20011003	eray	eray123	Eray	Gokce	987 Elm St	5558765432	era.gokce@example.com
5	20011004	michael	pass654	Michael	Jackson	654 Birch St	5557654321	michael.jackson@example.com
6	20011005	emily	pass321	Emily	White	321 Cedar St	5556543210	emily.white@example.com
7	20011006	ryan	pass012	Ryan	Martin	012 Maple St	5555432109	ryan.martin@example.com
8	20011007	laura	pass345	Laura	Davis	345 Walnut St	5554321098	laura.davis@example.com
9	20011008	brian	pass678	Brian	Harris	678 Pine St	5553210987	brian.harris@example.com
10	20011009	sophie	pass901	Sophie	Thomas	901 Oak St	5552109876	sophie.thomas@example.com

2. Ürün tablosu:

	urun_id [PK] integer	urun_isim character varying (100)	fiyat numeric (10,2)	satici_id integer	stok integer	satis_sayisi integer
1	1000	Laptop	2499.00	20011001	25	5
2	1001	Smartphone	899.00	20011002	49	1
3	1002	Smartwatch	199.00	20011003	17	3
4	1003	Tablet	599.00	20011001	14	1
5	1004	Wireless Earbuds	129.00	20011002	38	2
6	1005	Digital Camera	799.00	20011004	25	0
7	1006	Bluetooth Speaker	149.00	20011000	33	2
8	1007	Gaming Console	499.00	20011004	10	0
9	1008	Fitness Tracker	79.00	20011003	28	2
10	1009	4K TV	1499.00	20011000	4	1

3. Sipariş tablosu:

	siparis_id [PK] integer	siparis_tarihi date	alici_id integer	satici_id integer	urun_id integer	adet integer	toplam_ucret numeric (10,2)
1	5000	2024-01-13	20011004	20011001	1000	2	4998.00
2	5001	2024-01-13	20011005	20011002	1001	1	899.00
3	5002	2024-01-13	20011006	20011003	1002	3	597.00
4	5003	2024-01-13	20011007	20011001	1003	1	599.00
5	5004	2024-01-13	20011008	20011002	1004	2	258.00
6	5005	2024-01-13	20011009	20011003	1008	1	79.00
7	5006	2024-01-13	20011007	20011001	1000	3	7497.00
8	5007	2024-01-13	20011004	20011000	1009	1	1499.00
9	5008	2024-01-13	20011005	20011000	1006	2	298.00
10	5009	2024-01-13	20011006	20011003	1008	1	79.00

4. Satış tutarı tablosu

	satici_id integer	siparis_sayisi integer	total_satis numeric (10,2)
1	20011002	2	1157.00
2	20011001	3	13094.00
3	20011000	2	1797.00
4	20011003	3	755.00

Proje Maddeleri ve SQL Kod Blokları

1) **Oluşturacağınız veritabanı en az 4 tablo içermelidir**.

Bütün tabloları oluşturduğumuz SQL kod blokları :

1.tablo

```
CREATE SEQUENCE kullanici_id_seq START WITH 20011000;
CREATE TABLE kullanici (
 kullanici_id INTEGER DEFAULT nextval('kullanici_id_seq') PRIMARY KEY,
 kullanici_adi VARCHAR(50) UNIQUE NOT NULL,
 sifre VARCHAR(50) NOT NULL,
 ad VARCHAR(100),
 soyad VARCHAR(100),
 adres VARCHAR(255),
 telefon NUMERIC,
 email VARCHAR(55));
```

2.tablo

```
CREATE SEQUENCE urun_id_seq START WITH 1000;
CREATE TABLE urun (
 urun_id INTEGER DEFAULT nextval('urun_id_seq') PRIMARY KEY,
 urun_isim VARCHAR(100) NOT NULL,
 fiyat NUMERIC(10, 2) NOT NULL,
 satici_id INTEGER REFERENCES kullanici(kullanici_id),
 stok INTEGER NOT NULL,
 satis_sayisi INTEGER DEFAULT 0
);
```

3.tablo

```
CREATE SEQUENCE siparis_id_seq START WITH 5000;
CREATE TABLE Siparis (
 siparis_id INTEGER DEFAULT nextval('siparis_id_seq') PRIMARY KEY,
 siparis_tarihi DATE DEFAULT CURRENT_DATE,
 alici_id INTEGER REFERENCES kullanici(kullanici_id),
 satici_id INTEGER REFERENCES kullanici(kullanici_id),
 urun_id INTEGER REFERENCES urun(urun_id) ON DELETE CASCADE,
 adet INTEGER NOT NULL,
 toplam_ucret NUMERIC(10, 2)
);
```

4.tablo

```
CREATE TABLE satis_tutar(
satici_id INTEGER REFERENCES kullanici(kullanici_id),
siparis_sayisi INTEGER NOT NULL,
total_satis NUMERIC(10,2) NOT NULL
```

2) Tablolarınızda primary key ve foreign key kısıtlarını kullanmalısınız.

Aşağıdaki tabloda hem bu tabloya özgü bir primary key ve diğer tablolarla bağlı bir kısıt olan foreign key kullanılmıştır.

```
CREATE SEQUENCE siparis_id_seq START WITH 5000;
CREATE TABLE Siparis (
 siparis_id INTEGER DEFAULT nextval('siparis_id_seq') PRIMARY KEY,
 siparis_tarihi DATE DEFAULT CURRENT_DATE,
 alici_id INTEGER REFERENCES kullanici(kullanici_id),
 satici_id INTEGER REFERENCES kullanici(kullanici_id),
 urun_id INTEGER REFERENCES urun(urun_id) ON DELETE CASCADE ,
 adet INTEGER NOT NULL,
 toplam_ucret NUMERIC(10, 2)
);
```

3) En az bir tabloda silme kısıtı ve sayı kısıtı olmalıdır.

Aşağıdaki tabloyu oluştururken silme ve sayı kısıtı kullanılmıştır.

```
CREATE SEQUENCE siparis_id_seq START WITH 5000;
CREATE TABLE Siparis (
 siparis_id INTEGER DEFAULT nextval('siparis_id_seq') PRIMARY KEY,
 siparis_tarihi DATE DEFAULT CURRENT_DATE,
 alici_id INTEGER REFERENCES kullanici(kullanici_id),
 satici_id INTEGER REFERENCES kullanici(kullanici_id),
 urun_id INTEGER REFERENCES urun(urun_id) ON DELETE CASCADE ,
 adet INTEGER NOT NULL,
 toplam_ucret NUMERIC(10, 2)
);
```

4) Arayüzden en az birer tane insert, update ve delete işlemi gerçekleştirilebilmelidir.

Aşağıda farklı SQL kod bloklarında arayüzden alınan değerler ile insert, update ve delete işlemlerini gerçekleştiren 4 farklı sql kod bloğu vardır.

```
-- Kullanici Kayıt Olma
INSERT INTO kullanici(kullanici_adi, sifre , ad ,soyad , adres ,telefon, email )
VALUES (?, ? ,? ,? , ?, ? ,?);

--Kullanici Guncelle
UPDATE kullanici SET kullanici_adi='"+ kullaniciAdi + "', sifre = '" + yeniSifre + "'
, ad = '" + yeniAd + "', soyad = '" + yeniSoyad + "' , adres = '" + yeniAdres + "'
,telefon = '" + yeniTelefon + "', email = '" + yeniEmail + "' WHERE kullanici_id = '"
+ iD + "'  ";
```

```sql
-- Ürün Satın Alma Fonksiyonu --
CREATE OR REPLACE FUNCTION satin_al(p_alici_id INT, p_satici_id INT, p_urun_id INT,
p_adet INT)
RETURNS VOID AS $$
BEGIN
    -- Siparişi ekleyin
    INSERT INTO Siparis (alici_id, satici_id, urun_id, adet)
       VALUES (p_alici_id, p_satici_id, p_urun_id, p_adet);

    -- Stok ve satış sayısını güncelleyin
    UPDATE urun
    SET stok = stok - p_adet,
        satis_sayisi = satis_sayisi + p_adet
    WHERE urun_id = p_urun_id;

END;
$$ LANGUAGE 'plpgsql';

-- Urun Silme -- delete Product -- Fonksiyon Tanımlama
create or replace function deleteProduct(v_urun_id integer)
   returns integer as
$$
declare
begin
       DELETE FROM urun WHERE urun_id = v_urun_id;
       IF FOUND THEN
              return 1;
       ELSE
              return 0;
       end if;
end;
$$ language 'plpgsql';
```

5) Arayüzden girilecek bir değere göre ekrana sonuçların listelendiği bir sorgu yazmalısınız.

Arayüzden girilecek değere göre satıştaki ürünlerimizi gösteren SQL kod bloğu:

```sql
--Market Place -- Fiyata Gore Filtreleme
SELECT urun_isim,fiyat,satis_sayisi,stok FROM urun WHERE satici_id <> ? AND fiyat
BETWEEN ? AND ?;
```

6) Arayüzden çağrılan sorgulardan en az biri “view” olarak tanımlanmış olmalıdır.

```sql
--Saticinin Yaptığı sipariş adedi ve tüm siparişlerin toplam tutarı
gösteren(VIEW KULLANIMI)—

CREATE OR REPLACE VIEW satis_tutar_view AS SELECT satici_id, siparis_sayisi,
total_satis FROM satis_tutar;
```

```
SELECT *FROM satis_tutar_view INTERSECT SELECT
satici_id,siparis_sayisi,total_satis from satis_tutar where satici_id = ?;
```

7) En az bir adet “sequence” oluşturmalı ve arayüzden yapılacak insert sırasında ilgili sütundaki değerlerin otomatik olarak atanmasını sağlamalısınız.

Aşağıdaki kod bloğunda kullanıcı tablosu oluşturulurken kullanıcı_id sequence ile oluşturuldu ve arayüzden girilen kullanıcının diğer bilgilerinden sonra otomatik olarak atanmaktadır.

```
CREATE SEQUENCE kullanici_id_seq START WITH 20011000;
CREATE TABLE kullanici (
 kullanici_id INTEGER DEFAULT nextval('kullanici_id_seq') PRIMARY KEY,
 kullanici_adi VARCHAR(50) UNIQUE NOT NULL,
 sifre VARCHAR(50) NOT NULL,
 ad VARCHAR(100),
 soyad VARCHAR(100),
 adres VARCHAR(255),
 telefon NUMERIC,
 email VARCHAR(55));
```

8) Arayüzden çağrılan sorgulardan en az birinde union veya intersect veya except kullanmış olmalısınız.

Aşağıdaki kod bloğunda mağazadan ürün ismine göre filtreleme yaparken intersect kullanımını görüyoruz.

```
--My Market Place -- İsme Göre Filtreleme(INTERSECT KULLANIMI)
SELECT urun_isim, fiyat, satis_sayisi,stok FROM urun WHERE satici_id = ?;
INTERSECT
SELECT urun_isim, fiyat, satis_sayisi,stok FROM urun WHERE urun_isim LIKE '%?%';
```

9) Sorgularınızın en az biri aggregate fonksiyonlar içermeli, having ifadesi kullanılmalıdır.

Aşağıda total satış tutarını güncelleyen fonksiyonda aggregate fonksiyonları(count,sum) ve having ifadesini görüyoruz.

```
- Satın Alındıktan Sonra 4.Tablomuzdaki Total_Satış tutarımızı güncelleyen trigger
3(aggregate,Having KULLANIMI)--

CREATE OR REPLACE FUNCTION update_satis_tutar()
RETURNS TRIGGER AS $$
DECLARE
```

```
    satis_tutar_row satis_tutar%ROWTYPE;
BEGIN
    -- Satici_id'ye göre satis_tutar tablosunda ilgili satirin varlığını kontrol
edilir
    SELECT * INTO satis_tutar_row
    FROM satis_tutar
    WHERE satici_id = NEW.satici_id;

    -- Eğer satici_id'ye ait bir satir yoksa(satici daha once satıs yapmadıysa) yeni
bir satir ekleyin
    IF NOT FOUND THEN
        INSERT INTO satis_tutar (satici_id, siparis_sayisi, total_satis)
        VALUES (NEW.satici_id, 1, NEW.toplam_ucret);
    ELSE
        -- Eğer satir varsa, siparis_sayisi'ni artırın ve total_satis'i güncelleyin
        UPDATE satis_tutar
        SET siparis_sayisi = satis_tutar_row.siparis_sayisi + 1,
            total_satis = (SELECT SUM(toplam_ucret)
                           FROM Siparis
                           WHERE satici_id = NEW.satici_id
                           HAVING COUNT(siparis_id) > 1)
        WHERE satici_id = NEW.satici_id;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

10) Arayüzden girilen değerleri parametre olarak alıp ekrana sonuç döndüren **3 farklı SQL fonksiyonu** tanımlamış olmalısınız. Bu fonksiyonların en az birinde “record” ve “cursor” tanımı-kullanımı olmalıdır..

Aşağıda arayüzden girilen değerleri parametre olarak alıp ekrana sonuç döndüren 3 farklı SQL fonksiyonu var. İsme göre filtreleme yaparken record ve cursor kullanımını görüyoruz.

```
--My Market Place -- Stok Sayisina Gore Filtreleme
SELECT urun_isim,fiyat,satis_sayisi,stok FROM urun WHERE satici_id = ? AND stok
BETWEEN ? AND ?;

--My Market Place -- Fiyata Gore Filtreleme
SELECT urun_isim,fiyat,satis_sayisi,stok FROM urun WHERE satici_id = ? AND fiyat
BETWEEN ? AND ?;


--Market Place --İsme Göre Filtreleme Fonksiyonu Tanımı (RECORD VE CURSER KULLANIMI)
--Type Record Tanımı
CREATE TYPE urun_list AS (urun_isim varchar(100), fiyat NUMERIC, satis_sayisi INTEGER,
stok INTEGER);
CREATE OR REPLACE FUNCTION urun_sorgu(
    p_kullanici_id INTEGER,
    p_urun_isim varchar(20)
)
RETURNS SETOF urun_list AS $$
DECLARE
```

```
    urun_row urun_list;
    cur CURSOR FOR SELECT u.urun_isim, u.fiyat, u.satis_sayisi, u.stok FROM urun u
        WHERE u.satici_id <> p_kullanici_id
        AND u.urun_isim LIKE '%'  p_urun_isim  '%';
BEGIN
    FOR urun_row IN cur
    LOOP
        RETURN NEXT urun_row;
    END LOOP;
    RETURN;
END;
$$ LANGUAGE 'plpgsql';
```

11) 2 adet trigger tanımlamalı ve arayüzden girilecek değerlerle tetiklemelisiniz. Trigger'ın çalıştığına dair arayüze bilgilendirme mesajı döndürülmelidir.

```
----------Satın alım fonksiyonundan  etkilenen Trigger Fonksiyonları---------

-- Stok Kontrolü Yapan Trigger MESAJ DÖNDÜREN TRİGGER--TRIGGER 1
CREATE OR REPLACE FUNCTION stok_kontrol()
RETURNS TRIGGER AS $$
BEGIN

    IF NEW.adet > (SELECT stok FROM urun WHERE urun_id = NEW.urun_id) THEN
        RAISE EXCEPTION 'Stok yetersiz';
       ELSE
              RAISE NOTICE 'Stok kontrolü başarılı, sipariş eklenebilir';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE OR REPLACE TRIGGER before_siparis_insert
BEFORE INSERT ON Siparis
FOR EACH ROW
EXECUTE FUNCTION stok_kontrol();

--Satın Alımında Siparişin Satırındaki Toplam Tutarı Hesaplayan Trigger 2--
CREATE OR REPLACE FUNCTION toplam_fiyat()
RETURNS TRIGGER AS $$
BEGIN
  NEW.toplam_ucret = (SELECT fiyat FROM urun WHERE urun_id = NEW.urun_id) * NEW.adet;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_siparis
BEFORE INSERT ON Siparis
FOR EACH ROW
EXECUTE FUNCTION toplam_fiyat();
```