BIMU3064

Veritabanı Yönetim Sistemleri

ÖDEV 4

Abdulkadir Azmanoğlu
1306130092

1. Fis tablosundaki toplam alanını gerektiğinde güncelleyen trigger(lar)ı yazınız. İpucu: fisUrunleri tablosuna bir kayıt eklendiğinde, silindiğinde veya bir kayıttaki (barkod veya miktar veya birimFiyat alanları) guncellendiğinde çalışacak olan bu trigger fiş toplamı değerini fiş ürünlerindeki "miktar x birimFiyat" değerlerini toplayarak bulur.

```
CREATE OR REPLACE FUNCTION fisToplam () RETURNS TRIGGER AS $$
    BEGIN
        IF (TG_OP = 'INSERT') THEN
            UPDATE fis SET toplam = (
                                SELECT SUM(miktar * birimFiyat)
                                FROM fisUrunleri
                                WHERE fisNo = NEW.fisNo
                                )
                WHERE fisNo=NEW.fisNo;
            RETURN NEW;
        ELSIF (TG_OP = 'DELETE') THEN
            UPDATE fis SET toplam = (
                                SELECT SUM(miktar * birimFiyat)
                                FROM fisUrunleri
                                WHERE fisNo = OLD.fisNo
                                )
                WHERE fisNo=OLD.fisNo;
            RETURN OLD;
        ELSIF (TG_OP = 'UPDATE') THEN
            UPDATE fis SET toplam = (
                                SELECT SUM(miktar * birimFiyat)
                                FROM fisUrunleri
                                WHERE fisNo = OLD.fisNo
                                )
                WHERE fisNo=OLD.fisNo;
            RETURN NEW;
        END IF;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER fisToplamTrigger
AFTER INSERT OR UPDATE OR DELETE ON fisUrunleri FOR EACH ROW EXECUTE PROCEDURE fisToplam ();
```

2. Urun tablosundaki stokMiktarı alanını gerektiğinde güncelleyen trigger(lar)ı yazınız. İpucu: stokMiktarı bir ürünün tüm alımlardaki(fatura) miktarlar toplamından tüm satişlardaki miktarlar toplamının çıkarılmasıyla hesaplanabilir. Bu hesaplamadan birim alanı kullanılmaz.

```
CREATE OR REPLACE FUNCTION STOKMIKTAR () RETURNS TRIGGER AS $$
    BEGIN
        IF (TG_OP = 'INSERT') THEN
             UPDATE urun SET stokMiktari = (SELECT
                                (SELECT SUM(FAU.MIKTAR)
                                        FROM FATURAURUNLERI FAU
                                        WHERE FAU.BARKOD = NEW.barkod) -
                                (SELECT COALESCE(SUM(FIU.MIKTAR), 0)
                                        FROM FISURUNLERI FIU
                                        WHERE FIU.BARKOD = NEW.barkod) AS Stock)
                  WHERE barkod = NEW.barkod;
             RETURN NEW;
        ELSIF (TG_OP = 'DELETE') THEN
             UPDATE urun SET stokMiktari = (SELECT
                                (SELECT SUM(FAU.MIKTAR)
                                        FROM FATURAURUNLERI FAU
                                        WHERE FAU.BARKOD = OLD.barkod) -
                                (SELECT COALESCE(SUM(FIU.MIKTAR), 0)
                                        FROM FISURUNLERI FIU
                                        WHERE FIU.BARKOD = OLD.barkod) AS Stock)
                  WHERE barkod = OLD.barkod;
             RETURN OLD;
        ELSIF (TG_OP = 'UPDATE') THEN
             UPDATE urun SET stokMiktari = (SELECT
                                (SELECT SUM(FAU.MIKTAR)
                                        FROM FATURAURUNLERI FAU
                                        WHERE FAU.BARKOD = NEW.barkod) -
                                (SELECT COALESCE(SUM(FIU.MIKTAR), 0)
                                        FROM FISURUNLERI FIU
                                        WHERE FIU.BARKOD = NEW.barkod) AS Stock)
                  WHERE barkod = NEW.barkod;
             RETURN NEW;
        END IF;
    END;
$$ LANGUAGE PLPGSQL;


CREATE TRIGGER stokMiktarTrigger
AFTER INSERT OR UPDATE OR DELETE ON faturaUrunleri FOR EACH ROW EXECUTE PROCEDURE stokMiktar ();
CREATE TRIGGER stokMiktarTrigger
AFTER INSERT OR UPDATE OR DELETE ON fisUrunleri FOR EACH ROW EXECUTE PROCEDURE stokMiktar ();
```

3. Urun tablosundaki satisFiyatTrendi alanını gerektiğinde güncelleyen trigger(lar)ı yazınız. İpucu: satisFiyatTrendi alanı “artan” veya “azalan” olabilir. Bu değer ürünün tarihSaat’e göre son iki satışındaki (fisUrunlari) birimFiyat değerlerinin karşılaştırılmasıyla bulunur: Son satıştaki fiyat bir öncesinden büyükse trend “artan”, değilse “azalan” olacaktır.

```
CREATE OR REPLACE FUNCTION satisFiyatTrendi () RETURNS TRIGGER AS $$
        DECLARE
                x INTEGER;
                y INTEGER;
    BEGIN
        SELECT fiu.birimfiyat
                INTO x
        FROM FIS
        INNER JOIN fisUrunleri fiu ON fiu.fisno = fis.fisno
        WHERE barkod = NEW.barkod
        ORDER BY TARIHSAAT DESC
        LIMIT 1;
        SELECT fiu.birimfiyat
                INTO y
        FROM FIS
        INNER JOIN fisUrunleri fiu ON fiu.fisno = fis.fisno
        WHERE barkod = NEW.barkod
        ORDER BY TARIHSAAT DESC
        LIMIT 1
        OFFSET 1;

        UPDATE urun SET satisfiyattrendi = (case when x > y then 'Artan' when x < y then
'Azalan' else null end) WHERE barkod = NEW.barkod;

        RETURN NEW;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER satisTrendTrigger
AFTER INSERT OR UPDATE ON fisUrunleri FOR EACH ROW EXECUTE PROCEDURE satisFiyatTrendi ();
```

4. Fatura tablosundaki toplan alanını gerektiğinde güncelleyen trigger(lar)ı yazınız. İpucu: faturaUrunleri tablosuna bir kayıt eklendiğinde, silindiğinde veya bir kayıttaki (barkod veya miktar veya birimFiyat alanları) guncellendiğinde çalışacak olan bu trigger toplam değerini fatura ürünlerindeki "miktar x birimFiyat" değerlerini toplayarak bulur.

```
CREATE OR REPLACE FUNCTION faturaToplam () RETURNS TRIGGER AS $$
    BEGIN
        IF (TG_OP = 'INSERT') THEN
             UPDATE fatura SET toplam = (
                                SELECT SUM(miktar * birimFiyat)
                                FROM faturaUrunleri
                                WHERE faturaNo = NEW.faturaNo
                                )
                 WHERE faturaNo=NEW.faturaNo;
            RETURN NEW;
        ELSIF (TG_OP = 'DELETE') THEN
             UPDATE fatura SET toplam = (
                                SELECT SUM(miktar * birimFiyat)
                                FROM faturaUrunleri
                                WHERE faturaNo = OLD.faturaNo
                                )
                 WHERE faturaNo=OLD.faturaNo;
            RETURN OLD;
        ELSIF (TG_OP = 'UPDATE') THEN
             UPDATE fatura SET toplam = (
                                SELECT SUM(miktar * birimFiyat)
                                FROM faturaUrunleri
                                WHERE faturaNo = OLD.faturaNo
                                )
                 WHERE faturaNo=OLD.faturaNo;
            RETURN NEW;
        END IF;
    END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER faturaToplamTrigger
AFTER INSERT OR UPDATE OR DELETE ON faturaUrunleri FOR EACH ROW EXECUTE PROCEDURE faturaToplam
();
```

5. Tedarikçi tablosundaki faturaSayisi, faturaToplamı, faturaOrtalaması alanlarını gerektiğinde güncelleyen trigger(lar)ı yazınız. İpucu: fatura tablosuna kayıt eklenmesi, silinmesi veya bu tablodaki toplam değerinin güncellenmesi durumunda çalışacak bu trigger, soru 4'teki trigger'I dolaylı olarak kullanmalıdır. Çünkü faturaUrunleri tablosuna bir kayıt eklenmesi, silinmesi veya güncelleme durumunda tedarikci tablosundaki FaturaToplami ve faturaOrtalaması değerlerinin yeniden hesaplanması gerekecektir.

```
CREATE OR REPLACE FUNCTION tedarikciDetay () RETURNS TRIGGER AS $$
    BEGIN
        IF (TG_OP = 'INSERT') THEN
                UPDATE tedarikci SET faturaSayisi = (
                                SELECT COUNT(faturaNo)
                                FROM fatura
                                WHERE vergiNo = NEW.vergiNo
                        ),
                        faturaToplami = (
                                SELECT SUM(toplam)
                                FROM fatura
                                WHERE vergiNo = NEW.vergiNo
                        ),
                        faturaOrtalamasi = (
                                SELECT ROUND(AVG(toplam), 2)
                                FROM fatura
                                WHERE vergiNo = NEW.vergiNo
                        )
                WHERE vergiNo = NEW.vergiNo;
                RETURN NEW;
        ELSIF (TG_OP = 'DELETE') THEN
                UPDATE tedarikci SET faturaSayisi = (
                                SELECT COUNT(faturaNo)
                                FROM fatura
                                WHERE vergiNo = OLD.vergiNo
                        ),
                        faturaToplami = (
                                SELECT SUM(toplam)
                                FROM fatura
                                WHERE vergiNo = OLD.vergiNo
                        ),
                        faturaOrtalamasi = (
                                SELECT ROUND(AVG(toplam), 2)
                                FROM fatura
                                WHERE vergiNo = OLD.vergiNo
                        )
                WHERE vergiNo = OLD.vergiNo;
                RETURN OLD;
        ELSIF (TG_OP = 'UPDATE') THEN
                UPDATE tedarikci SET faturaSayisi = (
                                SELECT COUNT(faturaNo)
                                FROM fatura
                                WHERE vergiNo = OLD.vergiNo
                        ),
                        faturaToplami = (
                                SELECT SUM(toplam)
                                FROM fatura
                                WHERE vergiNo = OLD.vergiNo
                        ),
                        faturaOrtalamasi = (
                                SELECT ROUND(AVG(toplam), 2)
                                FROM fatura
                                WHERE vergiNo = OLD.vergiNo
                        )
                WHERE vergiNo = OLD.vergiNo;
                RETURN NEW;
        END IF;
    END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER tedarikciDetayTrigger
AFTER INSERT OR UPDATE OF toplam OR DELETE ON fatura FOR EACH ROW EXECUTE PROCEDURE
tedarikciDetay ();
```

6. "… satislar(barkod, tarihSaat) returns table…" gibi tanımlanan barkodu verilen bir ürünün verilen bir tarihSaatten sonraki tüm satış kayıtlarını (fisUrunlari tablosundaki kayıtları) döndüren stored function'I yazınız ve bu fonksiyonu çalıştığını gösterebilmek için bir SQL komutunda kullanınız. İpucu: "… satislar(barkod, tarihSaat) returns table…"

```
CREATE OR REPLACE FUNCTION sonrakiSatislar(barkodSorgu bigint, tarihSaatSorgu timestamp)
    RETURNS TABLE (
        fisNo INT,
        barkod BIGINT,
                miktar INT,
                birim TEXT,
                birimFiyat INT
)
AS $$
SELECT
        fiu.fisNo,
                fiu.barkod,
                fiu.miktar,
                fiu.birim,
                fiu.birimfiyat
    FROM
        fisUrunleri fiu
         INNER JOIN fis ON fis.fisNo = fiu.fisNo
    WHERE
        fiu.barkod = barkodSorgu AND fis.tarihsaat > tarihSaatSorgu;
$$
LANGUAGE  SQL STABLE;
```

```
SELECT * from sonrakiSatislar(4545435454, '31-12-2020 18:45:17');
```

fisno	barkod	miktar	birim	birimfiyat
53226	4545435454	2	Adet	7
53227	4545435454	2	Adet	9
53228	4545435454	2	Adet	14
53229	4545435454	2	Adet	16