SAKARYA
ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİTABANI YÖNETİM SİSTEMLERİ PROJE ÖDEVİ

Öğrenci Adı: Ali Koç

Öğrenci Numarası: G221210059

Ders Grubu: 2. Öğretim A Grubu

”Demiryolu Yönetim Sistemi” adlı bu uygulama modern bir demiryolu taşımacılığı yönetiminin verilerini düzenli bir şekilde tablolar halinde saklamak ve işlemek için tasarlanmıştır.

İş kuralları:

- Yolcular ve personel, kişiler tablosundan kalıtım alır.
- Bir kişi hem yolcu hem personel olabilir, ama en az biri olmak zorundadır.
- Bir kişinin isim soyisim bilgileri mevcuttur.
- Bir konumun ülke adı, şehir adı ve ilçe adına sahip olması zorunludur.
- Bir istasyonun konumu ve istasyon adı bulunur
- Bir personel sadece bir istasyonda çalışabilir, bir istasyonda birden çok personel çalışabilir.
- Bir istasyon sadece bir konumda bulunabilir, bir konumda birden çok istasyon bulunabilir.
- Bir rotanın başlangıç istasyonu ve hedef istasyonu mevcuttur.
- Bir trenin adı, kapasitesi, hızı ve türü mevcuttur.
- Belirli bir tren için bakım yapıldığında bakım türü ve tarihi kayıt edilir.
- Bir tren için birçok kez bakım yapılabilir, bir bakım sadece bir trene aittir
- Herhangi bir tren için durum tablosu oluşturulur; gidilen toplam mesafe, taşınan toplam yolcu, kazanılan toplam para kayıt edilir.
- Bir trenin sadece bir durum tablosu olabilir, bu tablo sadece bir trene ait olabilir.
- Yolculuklar; trenin numarası, takip edilecek rota ve kalkış tarihi ile kayıt edilir.
- Bir tren birden çok yolculuk yapabilir, bir yolculuk sadece o trene aittir.
- Herhangi bir yolculuğun gecikmesi durumunda yolculuk numarası ile beraber gecikme süresi ve gecikme sebebi kayıt edilir.
- Bir yolculuk birden fazla kez gecikebilir, ama bir gecikme sadece bir yolculuğa ait olabilir.
- Peronlar belirli bir istasyonda bulunur, istasyonNo ve trenNo kayıt edilir.
- Bir tren sadece bir peronda bulunabilir, bir peronda sadece bir tren bulunabilir.
- Bir istasyonda birden çok peron bulunabilir, bir peron sadece bir istasyonda bulunur.
- Duraklar için istasyonNo, yolculukNo, kalkış saati ve varış saati bilgileri mevcuttur.
- Bir durak sadece bir istasyonda bulunabilir, bir istasyonda birden çok durak bulunabilir.
- Bir yolculukta birden çok kez durak olabilir, ama bir durak sadece bir yolculukta gerçekleşir.
- Biletler için yolcu numarası, yolculuk numarası, kolktuk numarası ve ücret bilgileri mevcuttur.
- Bir yolcu birden çok bilet alabilir, ama bir bilet sadece bir yolcuya aittir.
- Bir yolculuk için birden çok bilet bulunabilir, ama bir bilet sadece bir yolculuğa aittir.
- Biletlerin satışından elde edilen kazanç, kazanç tablosunda saklanır.
- Yolcu tarafından herhangi bir bilet için geri bildirim verilebilir; bilet numarası, verilen puan ve yorum kayıt edilir.
- Belli bir yolcuya ait bilet için sadece bir yorum yapılabilir, bir yorum sadece belli bir bilete ait olabilir.

İlişkisel Şema:

konumlar(**konumNo: serial**, ulkeAdi: varchar(50), sehirAdi: varchar(50), ilceAdi: varchar(50))

istasyonlar(**istasyonNo: serial**, istasyonAdi: varchar(50), konumNo: integer)

rotalar(**rotaNo: serial**, baslangicIstasyonNo: integer, hedefIstasyonNo: integer)

kisiler(**kisiNo: serial**, isim: varchar(50), soyisim: varchar(50), yolcuMu: bool, personelMi: bool)

yolcular(**kisiNo: integer**, telefon: varchar(14), eposta: varchar(50), kazanilanPuan: integer)

personel(**kisiNo: integer**, istasyonNo: integer, pozisyon: varchar(50), maas: integer)

trenler(**trenNo: serial**, trenAdi: varchar(50), trenKapasitesi: integer, trenHiziKmh: integer, trenTuru: varchar(50))

trendurumu(**trenNo: integer**, gidilenToplamMesafe: integer, tasinanToplamYolcu: integer, kazanilanToplamPara: integer)

trenbakimi(**bakimNo: serial**, trenNo: integer, bakimTuru: varchar(50), bakimTarihi: date)

yolculuklar(**yolculukNo: serial**, trenNo: integer, rotaNo: integer, kalkisTarihi: timestamp, varisTarihi: timestamp)

gecikmeler(**gecikmeNo: serial**, yolculukNo: integer, gecikmeSuresi: interval)

peronlar(**peronNo: serial**, istasyonNo: integer, trenNo: integer)

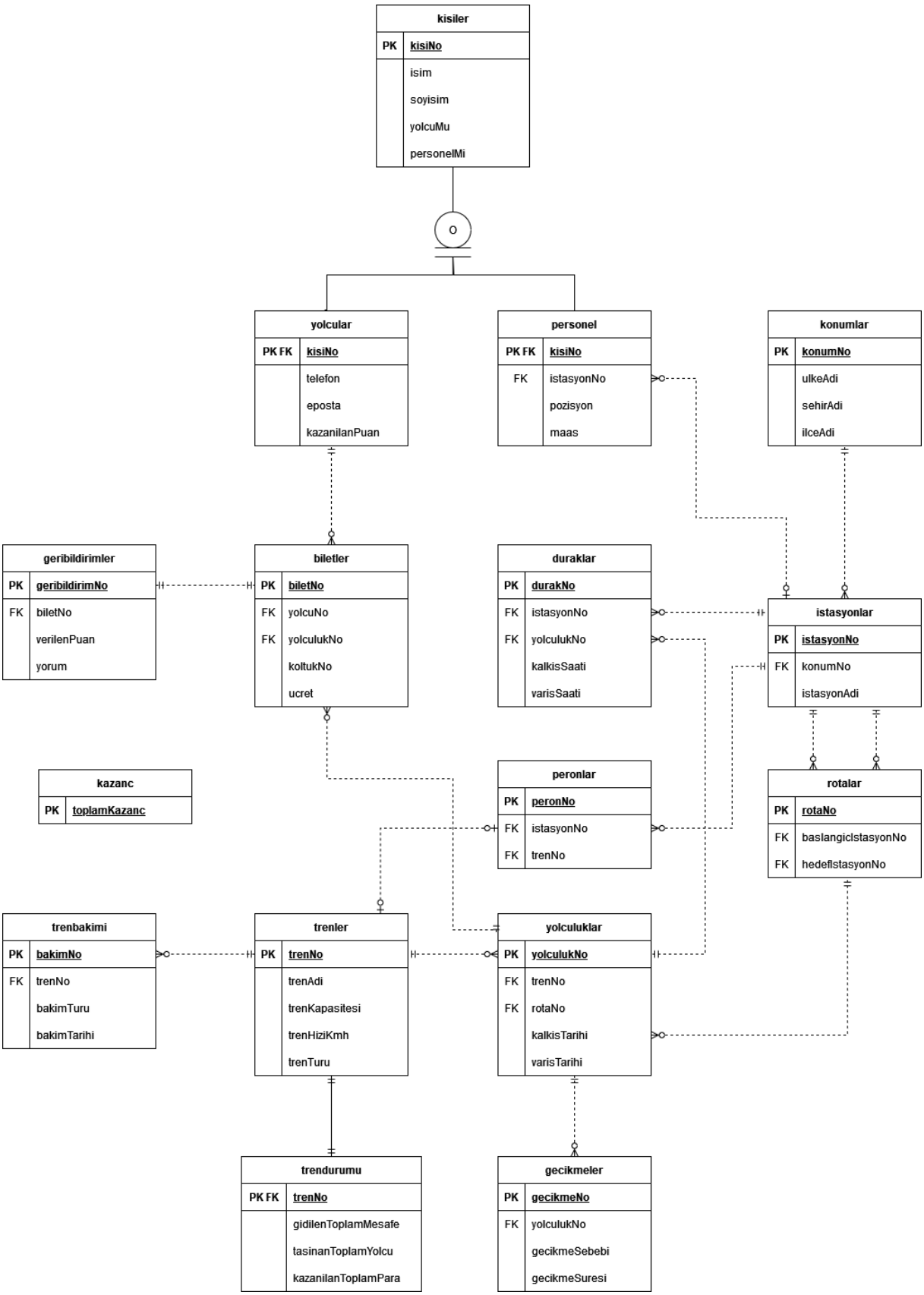
kazanc(**toplamKazanc: integer**)

duraklar(**durakNo: serial**, istasyonNo: integer, yolculukNo: integer, kalkisSaati:time, varisSaati:time)

biletler(**biletNo: serial**, yolcuNo: integer, yolculukNo: integer, koltukNo: integer, ucret: integer)

geribildirimler(geribildirimNo: serial, biletNo: integer, verilenPuan: integer, yorum: text)

Varlık Bağıntı Modeli:



SQL ifadeleri:

```
CREATE TABLE konumlar (
    konumNo SERIAL PRIMARY KEY,
    ulkeAdi VARCHAR(50) NOT NULL,
    sehirAdi VARCHAR(50) NOT NULL,
    ilceAdi VARCHAR(50) NOT NULL
);

CREATE TABLE istasyonlar (
    istasyonNo SERIAL PRIMARY KEY,
    istasyonAdi VARCHAR(50) NOT NULL,
    konumNo INTEGER NOT NULL REFERENCES konumlar(konumNo) ON DELETE CASCADE
);

CREATE TABLE rotalar (
    rotaNo SERIAL PRIMARY KEY,
    baslangicIstasyonNo INTEGER NOT NULL REFERENCES istasyonlar(istasyonNo) ON DELETE
CASCADE,
    hedefIstasyonNo INTEGER NOT NULL REFERENCES istasyonlar(istasyonNo) ON DELETE CASCADE
    CONSTRAINT baslangicVeHedefAyniOlamaz UNIQUE (baslangicIstasyonNo, hedefIstasyonNo)
);

CREATE TABLE kisiler (
    kisiNo SERIAL PRIMARY KEY,
    isim VARCHAR(50) NOT NULL,
    soyisim VARCHAR(50) NOT NULL,
    yolcuMu BOOLEAN NOT NULL,
    personelMi BOOLEAN NOT NULL
);

CREATE TABLE yolcular (
    kisiNo INTEGER PRIMARY KEY REFERENCES kisiler(kisiNo) ON UPDATE CASCADE ON DELETE
CASCADE,
    telefon VARCHAR(14),
    eposta VARCHAR(50),
    kazanilanPuan INTEGER DEFAULT 0
);
```

```
CREATE TABLE personel (
    kisiNo INTEGER PRIMARY KEY REFERENCES kisiler(kisiNo) ON UPDATE CASCADE ON DELETE
CASCADE,
    istasyonNo INTEGER REFERENCES istasyonlar(istasyonNo) ON DELETE SET NULL,
    pozisyon VARCHAR(50),
    maas INTEGER
);

CREATE TABLE trenler (
    trenNo SERIAL PRIMARY KEY,
    trenAdi VARCHAR(50) NOT NULL,
    trenKapasitesi INTEGER,
    trenHiziKmh INTEGER,
    trenTuru VARCHAR(50)
);

CREATE TABLE trendurumu (
    trenNo INTEGER PRIMARY KEY REFERENCES trenler(trenNo) ON UPDATE CASCADE ON DELETE
CASCADE,
    gidilenToplamMesafe INTEGER,
    tasinanToplamYolcu INTEGER,
    kazanilanToplamPara INTEGER
);

CREATE TABLE trenbakimi (
    bakimNo SERIAL PRIMARY KEY,
    trenNo INTEGER NOT NULL REFERENCES trenler(trenNo) ON DELETE CASCADE,
    bakimTuru VARCHAR(50),
    bakimTarihi DATE
);

CREATE TABLE yolculuklar (
    yolculukNo SERIAL PRIMARY KEY,
    trenNo INTEGER NOT NULL REFERENCES trenler(trenNo) ON DELETE CASCADE,
    rotaNo INTEGER NOT NULL REFERENCES rotalar(rotaNo) ON DELETE CASCADE,
    kalkisTarihi TIMESTAMP,
    varisTarihi TIMESTAMP
);
```

```
CREATE TABLE gecikmeler (
    gecikmeNo SERIAL PRIMARY KEY,
    yolculukNo INTEGER NOT NULL REFERENCES yolculuklar(yolculukNo) ON DELETE CASCADE,
    gecikmeSuresi INTERVAL
);

CREATE TABLE peronlar (
    peronNo SERIAL PRIMARY KEY,
    istasyonNo INTEGER NOT NULL REFERENCES istasyonlar(istasyonNo) ON DELETE CASCADE,
    trenNo INTEGER NOT NULL REFERENCES trenler(trenNo) ON DELETE CASCADE
);

CREATE TABLE duraklar (
    durakNo SERIAL PRIMARY KEY,
    istasyonNo INTEGER NOT NULL REFERENCES istasyonlar(istasyonNo) ON DELETE CASCADE,
    yolculukNo INTEGER NOT NULL REFERENCES yolculuklar(yolculukNo) ON DELETE CASCADE,
    kalkisSaati TIME,
    varisSaati TIME
);

CREATE TABLE biletler (
    biletNo SERIAL PRIMARY KEY,
    yolcuNo INTEGER NOT NULL REFERENCES yolcular(kisiNo) ON DELETE CASCADE,
    yolculukNo INTEGER NOT NULL REFERENCES yolculuklar(yolculukNo) ON DELETE CASCADE,
    koltukNo INTEGER NOT NULL,
    ucret INTEGER,
    CONSTRAINT yolculukVeKoltukAyniOlamaz UNIQUE (yolculukNo, koltukNo)
);

CREATE TABLE kazanc (
    toplamKazanc INTEGER PRIMARY KEY
);

CREATE TABLE geribildirimler (
    geribildirimNo SERIAL PRIMARY KEY,
    biletNo INTEGER REFERENCES biletler(biletNo) ON DELETE CASCADE,
    verilenPuan INTEGER,
    yorum TEXT
```

```
);

CREATE OR REPLACE FUNCTION ortalamaPuanHesapla(geciciYolculukNo INTEGER)
RETURNS NUMERIC
AS
$$
DECLARE
    geciciOrtalamaPuan NUMERIC;
BEGIN
    SELECT AVG(verilenPuan) INTO geciciOrtalamaPuan FROM geribildirimler
    WHERE biletNo IN (SELECT biletNo FROM biletler WHERE yolculukNo = geciciYolculukNo);

    RETURN COALESCE(geciciOrtalamaPuan, 0);
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION ortalamaGecikmeHesapla()
RETURNS INTERVAL
AS
$$
DECLARE
    ortalamaGecikme INTERVAL;
BEGIN
    SELECT AVG(gecikmeSuresi) INTO ortalamaGecikme FROM gecikmeler;
    RETURN ortalamaGecikme;
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION enKazancliTren()
RETURNS TABLE (trenNo INTEGER, trenAdi VARCHAR(50))
AS
$$
BEGIN
    RETURN QUERY EXECUTE
        'SELECT trenler.trenNo, trenler.trenAdi FROM trenler
        JOIN trendurumu on trenler.trenNo = trendurumu.trenNo
```

```
        ORDER BY trendurumu.kazanilanToplamPara DESC
        LIMIT 1';
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION bosKoltukSayisi(geciciYolculukNo INTEGER)
RETURNS INTEGER
AS
$$
DECLARE
    toplamKoltuk INTEGER;
    doluKoltuk INTEGER;
    bosKoltuk INTEGER;
BEGIN
    SELECT trenKapasitesi INTO toplamKoltuk FROM trenler
    WHERE trenNo = (SELECT trenNo FROM yolculuklar WHERE yolculukNo = geciciYolculukNo);

    SELECT COUNT(*) INTO doluKoltuk FROM biletler
    WHERE yolculukNo = geciciYolculukNo;

    bosKoltuk := toplamKoltuk - doluKoltuk;
    RETURN bosKoltuk;
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE yolcuyaPuanVer(geciciYolcuNo INTEGER, verilecekPuan INTEGER)
AS
$$
BEGIN
    UPDATE yolcular
    SET kazanilanPuan = kazanilanPuan + verilecekPuan
    WHERE kisiNo = geciciYolcuNo;
END;
$$
LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION kisiUygunluk()
RETURNS TRIGGER
AS
$$
BEGIN
    IF NEW.yolcuMu = false AND NEW.personelMi = false THEN
        RAISE EXCEPTION 'Kisi yolcu ya da personel, ikisinden biri olmak zorunda';
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER kisiUygunlukTrigger
BEFORE INSERT ON kisiler
FOR EACH ROW
EXECUTE FUNCTION kisiUygunluk();

CREATE OR REPLACE FUNCTION yolcuUygunluk()
RETURNS TRIGGER
AS
$$
BEGIN
    IF (SELECT yolcuMu FROM kisiler WHERE kisiNo = NEW.kisiNo) = false THEN
        RAISE EXCEPTION 'Eklenmeye calisilan kisi yolcu degil, sadece yolcu eklenebilir';
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER yolcuUygunlukTrigger
BEFORE INSERT ON yolcular
FOR EACH ROW
EXECUTE FUNCTION yolcuUygunluk();
```

```
CREATE OR REPLACE FUNCTION personelUygunluk()
RETURNS TRIGGER
AS
$$
BEGIN
    IF (SELECT personelMi FROM kisiler WHERE kisiNo = NEW.kisiNo) = false THEN
        RAISE EXCEPTION 'Eklenmeye calisilan kisi personel degil, sadece personel
eklenebilir';
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER personelUygunlukTrigger
BEFORE INSERT ON personel
FOR EACH ROW
EXECUTE FUNCTION personelUygunluk();

CREATE OR REPLACE FUNCTION biletAlinincaTrenDurumuGuncelle()
RETURNS TRIGGER
AS
$$
BEGIN
    UPDATE trendurumu
    SET tasinanToplamYolcu = tasinanToplamYolcu + 1,
        kazanilanToplamPara = kazanilanToplamPara + NEW.ucret
    WHERE trenNo = (SELECT trenNo FROM yolculuklar WHERE yolculukNo = NEW.yolculukNo);

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER biletAlinincaTrenDurumuGuncelleTrigger
AFTER INSERT ON biletler
FOR EACH ROW
```

```
EXECUTE FUNCTION biletAlinincaTrenDurumuGuncelle();

CREATE OR REPLACE FUNCTION biletSilinincaTrenDurumuGuncelle()
RETURNS TRIGGER
AS
$$
BEGIN
    UPDATE trendurumu
    SET tasinanToplamYolcu = tasinanToplamYolcu - 1,
        kazanilanToplamPara = kazanilanToplamPara - OLD.ucret
    WHERE trenNo = (SELECT trenNo FROM yolculuklar WHERE yolculukNo = OLD.yolculukNo);

    RETURN OLD;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER biletSilinincaTrenDurumuGuncelleTrigger
AFTER DELETE ON biletler
FOR EACH ROW
EXECUTE FUNCTION biletSilinincaTrenDurumuGuncelle();

CREATE OR REPLACE FUNCTION yolculukGecikinceGuncelle()
RETURNS TRIGGER
AS
$$
DECLARE
    geciciGecikmeSuresi INTERVAL;
BEGIN
    SELECT gecikmeSuresi INTO geciciGecikmeSuresi FROM gecikmeler
    WHERE yolculukNo = NEW.yolculukNo;

    UPDATE yolculuklar
    SET varisTarihi = varisTarihi + geciciGecikmeSuresi
    WHERE yolculukNo = NEW.yolculukNo;

    RETURN NEW;
END;
```

```
$$
LANGUAGE plpgsql;

CREATE TRIGGER yolculukGecikinceGuncelleTrigger
AFTER INSERT ON gecikmeler
FOR EACH ROW
EXECUTE FUNCTION yolculukGecikinceGuncelle();

CREATE OR REPLACE FUNCTION kazanciHesapla()
RETURNS TRIGGER
AS
$$
BEGIN
    UPDATE kazanc
    SET toplamKazanc = (SELECT SUM(kazanilanToplamPara) FROM trendurumu);

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER kazanciHesaplaTrigger
AFTER INSERT OR UPDATE OR DELETE ON trendurumu
FOR EACH ROW
EXECUTE FUNCTION kazanciHesapla();

INSERT INTO kazanc (toplamKazanc) VALUES (0);

INSERT INTO konumlar (ulkeAdi, sehirAdi, ilceAdi) VALUES
    ('Türkiye', 'Kocaeli', 'İzmit'),
    ('Türkiye', 'İstanbul', 'Kadıköy'),
    ('İngiltere', 'Londra', 'Greenwich');

INSERT INTO istasyonlar (istasyonAdi, konumNo) VALUES
    ('İzmit İstasyonu', 1),
    ('Kadıköy İstasyonu', 2),
    ('Greenwich İstasyonu', 3);
```

```
INSERT INTO rotalar (baslangicIstasyonNo, hedefIstasyonNo) VALUES
    (1, 2),
    (2, 3),
    (3, 1);

INSERT INTO kisiler (isim, soyisim, yolcuMu, personelMi) VALUES
    ('Arda', 'Demir', true, false),
    ('Elif', 'Çelik', true, false),
    ('Zeynep', 'Yılmaz', false, true),
    ('Burak', 'Aksoy', false, true),
    ('Ali', 'Koç', true, true);

INSERT INTO yolcular (kisiNo, telefon, eposta, kazanilanPuan) VALUES
    (1, '05357846524', 'ardad@example.com', 10),
    (2, '05357824158', 'c_elif@example.com', 20),
    (5, '05456884258', 'koc_ali@example.com', 30);

INSERT INTO personel (kisiNo, istasyonNo, pozisyon, maas) VALUES
    (3, 3, 'Platform görevlisi', 1500),
    (4, 2, 'Bilet satış görevlisi', 2000),
    (5, 1, 'Bakım teknisyeni', 2000);

INSERT INTO trenler (trenAdi, trenKapasitesi, trenHiziKmh, trenTuru) VALUES
    ('A-treni', 400, 150, 'Yolcu treni'),
    ('B-treni', 350, 180, 'Yolcu treni'),
    ('C-treni', 125, 400, 'Maglev');

INSERT INTO trendurumu (trenNo, gidilenToplamMesafe, tasinanToplamYolcu,
kazanilanToplamPara) VALUES
    (1, 824000, 6870, 1717500),
    (2, 225000, 2315, 578750),
    (3, 374000, 1892, 473000);

INSERT INTO trenbakimi (trenNo, bakimTuru, bakimTarihi) VALUES
    (1, 'Ray inceleme ve onarım', '2024-03-05'),
    (2, 'Filtre değişimi', '2024-04-08'),
    (3, 'Fren parça değişimi', '2025-04-16'),
    (1, 'Günlük bakım', '2024-04-17'),
```

```
    (1, 'Günlük bakım', '2024-04-18'),
    (1, 'Günlük bakım', '2024-04-19');

INSERT INTO yolculuklar (trenNo, rotaNo, kalkisTarihi, varisTarihi) VALUES
    (1, 1, '2024-05-01 11:00:00', '2024-05-01 14:00:00'),
    (2, 2, '2024-05-04 12:25:00', '2024-05-04 15:25:00'),
    (2, 3, '2024-05-11 12:30:00', '2024-05-11 14:30:00'),
    (3, 3, '2024-05-15 14:00:00', '2024-05-15 17:30:00');

INSERT INTO gecikmeler (yolculukNo, gecikmeSuresi) VALUES
    (1, '2 hours'),
    (2, '5 hours'),
    (3, '3 hours'),
    (4, '6 hours');

INSERT INTO peronlar (istasyonNo, trenNo) VALUES
    (1, 3),
    (2, 2),
    (3, 1);

INSERT INTO duraklar (istasyonNo, yolculukNo, kalkisSaati, varisSaati) VALUES
    (1, 2, '17:25:00', '20:45:00'),
    (2, 3, '15:30:00', '19:20:00');

INSERT INTO biletler (yolcuNo, yolculukNo, koltukNo, ucret) VALUES
    (1, 1, 15, 250),
    (1, 2, 32, 250),
    (2, 2, 43, 250),
    (5, 4, 10, 250);

INSERT INTO geribildirimler (biletNo, verilenPuan, yorum) VALUES
    (1, 3, 'Sadece 2 saat gecikti, beklediğimden iyi bir performans'),
    (2, 1, '3 saatlik yol için 5 saat beklemek zorunda kaldım'),
    (3, 5, 'Hayatımın en iyi yolculuğuydu'),
    (4, 4, 'Gayet normal');
```

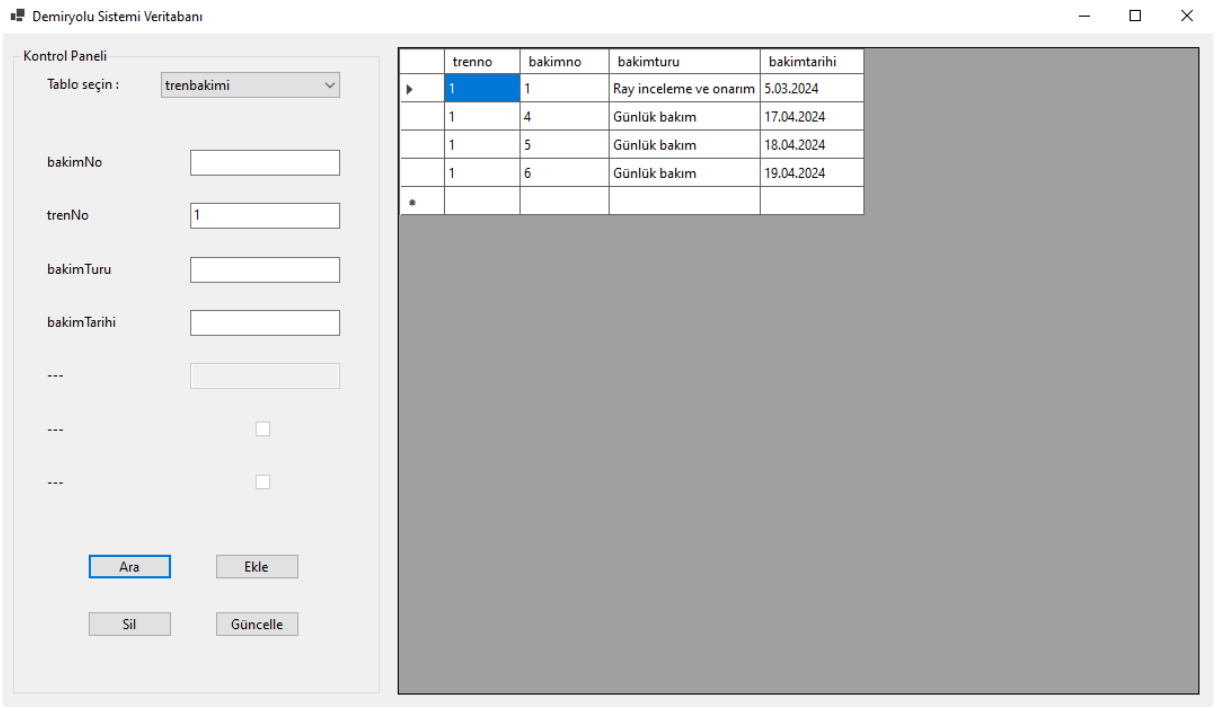
Saklı Yordam – Fonksiyonlar:

- bosKoltukSayisi(geciciYolculukNo integer): Parametre olarak girilen yolculuğa ait trenin kapasitesinden dolu olan koltuk miktarını çıkararak trende kaç koltuğun boş olduğunu gösteriyor.
- enKazancliTren(): Trendurumu tablosu üzerinden en çok para kazanan treni döndürüyor.
- ortalamaGecikmeHesapla(): Gecikmeler tablosundaki gecikme sürelerinin ortalamasını döndürüyor.
- ortalamaPuanHesapla(geciciYolculukNo integer): Parametre olarak girilen yolculuk için yolcuların verdiği puanların ortalamasını döndürüyor.
- yolcuyaPuanVer(IN geciciYolcuNo integer, IN verilecekPuan integer): Parametre olarak girilen yolcunun hesabına parametre üzerinden verilen miktar kadar puan ekliyor.

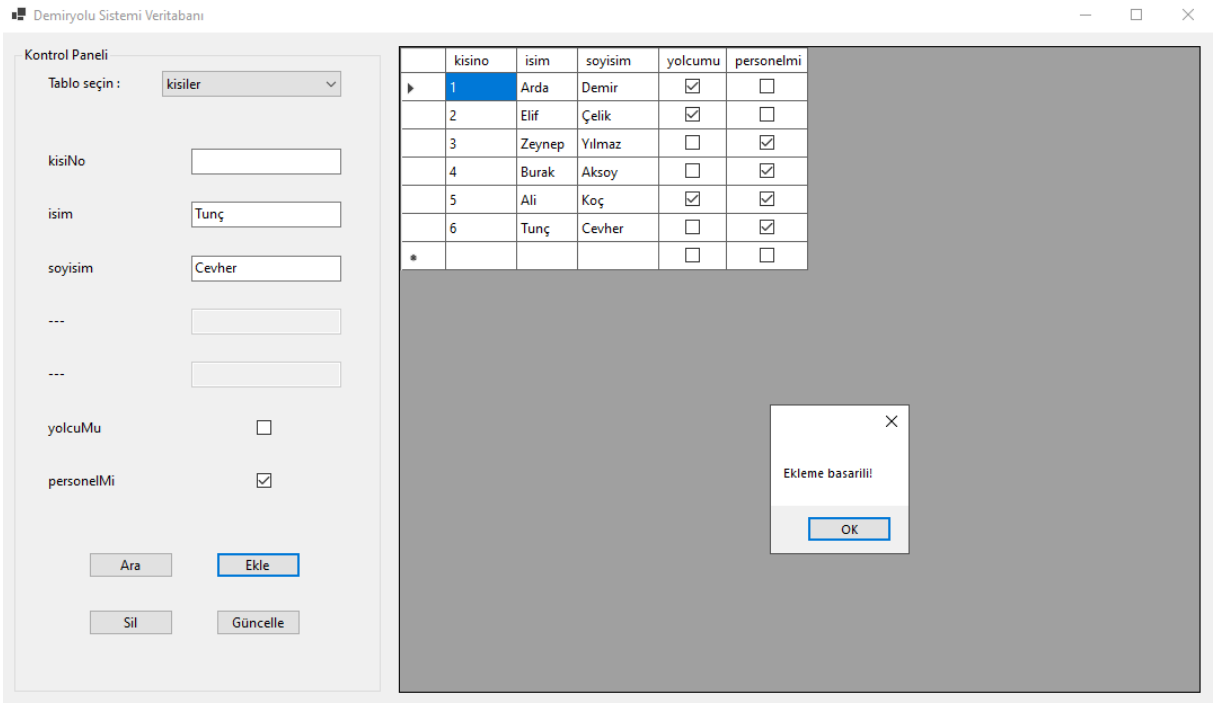
Tetikleyiciler:

- biletAlinincaTrendurumuGuncelle(): Biletler tablosuna yeni bilet eklendiğinde, ait olduğu yolculuğu yapan trenin trendurumu tablosu güncelleniyor (taşıdığı yolcu sayısı ve kazandığı para artıyor).
- biletSilininceTrendurumuGuncelle(): Biletler tablosundan bir bilet silinirse silinen biletin trendurumu tablosuna yaptığı etkisi geri çekiliyor.
- kazanciHesapla(): Trendurumu tablosunda meydana gelen herhangi bir değişiklikten sonra trenlerin toplam kazanç miktarı kazanc tablosuna kaydediliyor.
- kisiUygunluk(): Kisiler tablosuna yeni bir kişi eklenmeye çalışıldığında bu eklemenin uygunluğu kontrol ediliyor (yolcuMu ve personelMi değerleri aynı anda false olamaz).
- personelUygunluk():Personel tablosuna yeni bir kişi eklenmeye çalışıldığında bu eklemenin uygunluğu kontrol ediliyor (personelMi değeri true olmak zorunda).
- yolcuUygunluk(): Yolcular tablosuna yeni bir kişi eklenmeye çalışıldığında bu eklemenin uygunluğu kontrol ediliyor (yolcuMu değer true olmak zorunda).
- yolculukGecikinceGuncelle(): Eğer belli bir yolculuk için gecikme yaşanırsa, yolculuk tablosundaki varisTarihi değeri gecikme süresi kadar erteleniyor.

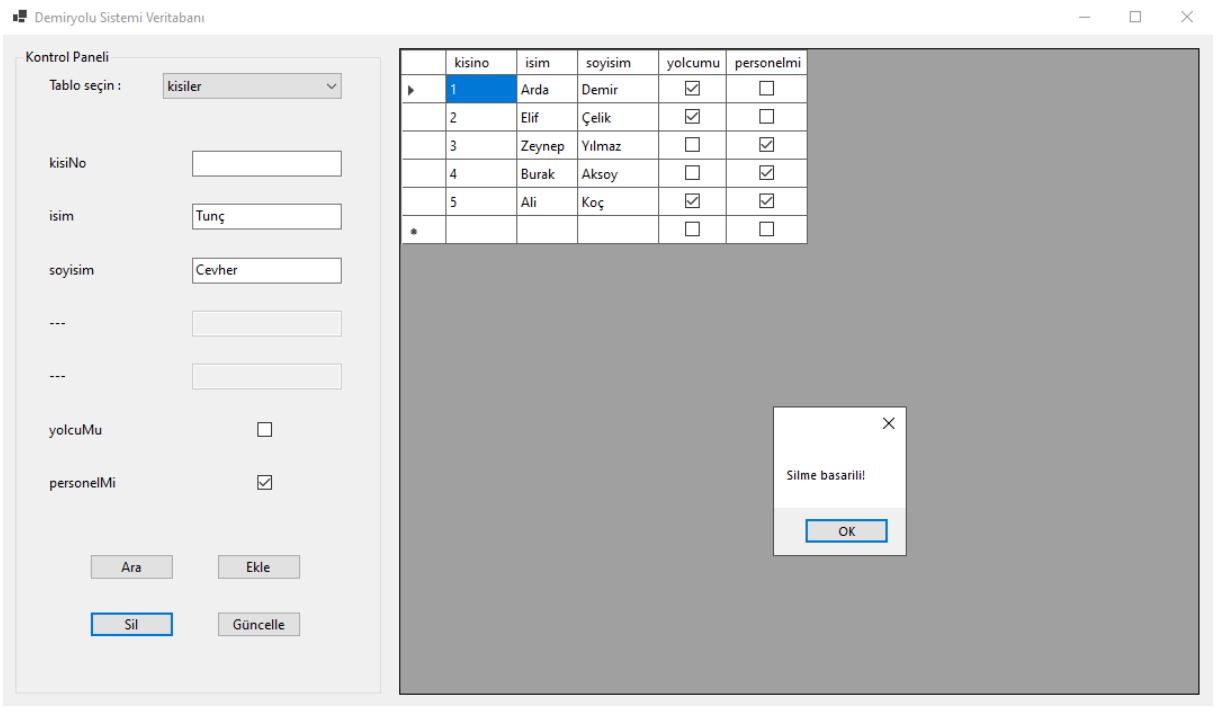
Ekran Görüntüleri:



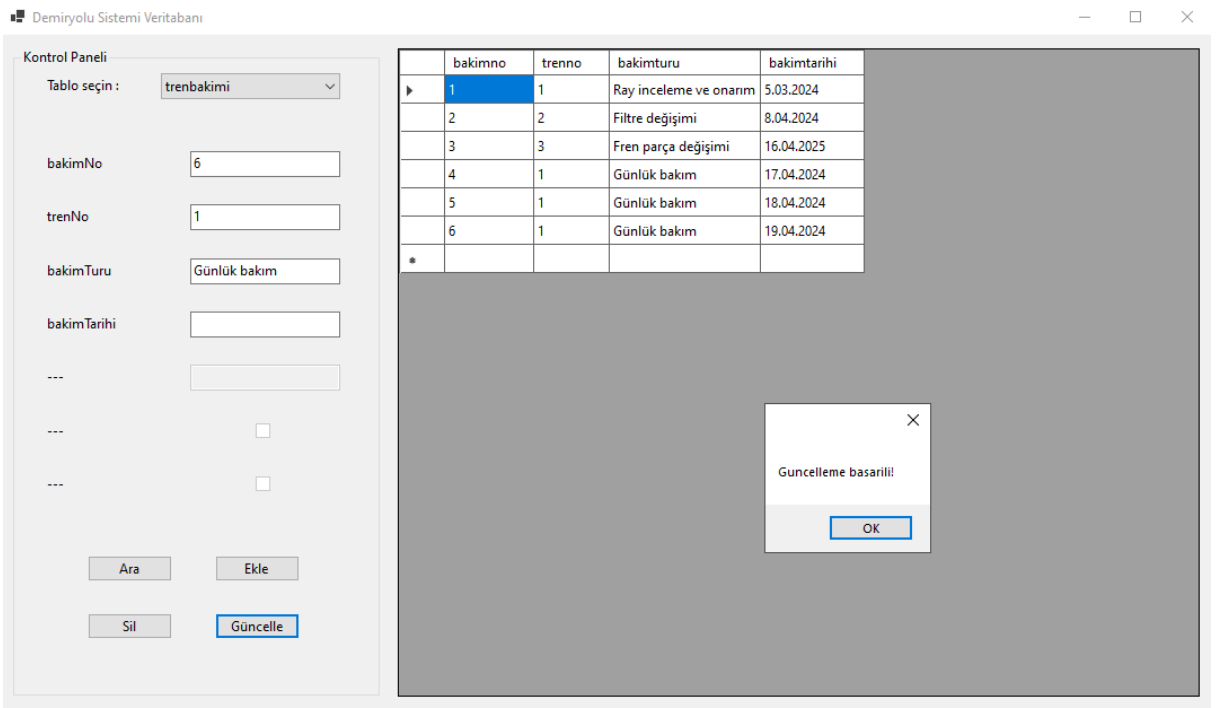
Arama işlemi



Ekleme işlemi



Silme işlemi



Güncelleme işlemi

Kaynak dosyalarının bulunduğu Github linki:

https://github.com/koc-ali88/Demiryolu-yonetim-sistemi