2. Notlandırma - Proje Gereksinimlerinin Belirlenmesi ve E-R Diyagramlarının Oluşturulması

Notlandırma tarihleri: 5 Kasım (I.Öğretim), 6 Kasım (II. Öğretim)

2. Notlandırmada dikkat edilecek hususlar:

- Proje grupları dönem projeleri için **github sayfası oluşturacaklardır**. Her grup proje grupları listesinden grup numarasına bakıp linkteki listeye proje için oluşturdukları github sayfasının adresini eklemelidir.
- 2.Notlandırma için **proje gereksinimleri ve E-R diyagramı** aşağıdaki belirlenen formata göre oluşturulacak ve her grup **github sayfasına ekleyecektir**, notlandırma github sayfası üzerinden yapılacaktır.
- 2. Notlandırma yukarıda belirtilen tarihlerde lab. saatinde yapılacak, öğrenciler proje gereksinimlerini ve E-R diyagramlarını github sayfalarından açıp anlatacaklardır.

BMÜ329 Veri Tabanı Sistemleri Dersi Dönem Projesi Gereksinimleri ve E-R Diyagramı Formatı

Proje Başlığı: Araç Otopark Otomasyonu
Proje Ekibindeki Kişiler: Veysel Emre Tay – Onur Ergün – Enes Argama

ÖNEMLİ: İlişkisel Veri Tabanı Yönetim Sistemlerinin temel pratiklerini uygulayabilmek için projelerde **varlıklar (tablolar) arası ilişkilerin zengin olması** beklenmektedir. Örneğin bir e-ticaret veritabanı projesi için "Müşterilerin şimdiye yapmış oldukları siparişlerde hangi üründen kaç adet aldığı listelenecek" şeklindeki bir gereksinim müşteri, sipariş, sipariş detay ve ürün tabloları arasında ilişki olmasını gerektirir. Proje konuları buna benzer senaryolara uygun değilse öğrenciler farklı bir proje konuları belirleyebilirler.

1- Dönem Projesi Gereksinimleri

Proje grupları veri tabanı dönem projelerinin gereksinimleri detaylı bir şekilde açıklanacaktır. Projede olması gereken özellikler yapılacak işlemler açıklanacaktır. Projede hangi varlıkların olacağı bu varlıkların niteliklerinin neler olacağı, bu varlıklar arasında ne gibi ilişkiler olacağı, ilişkilerde ne gibi sayısal kısıtlamalar (1:1,1:n, n:m vb.) olacağı (örneğin bir müşteri birden fazla sipariş verebilecek, ancak her sipariş tek bir müşteri tarafından yapılabilecek), varlıkların nitelikleri ile ilgili ne gibi kısıtlamalar olacağı açıklayacaklardır.

Gereksinimler farklı kullanıcı türlerine göre ve projenin konusunu kapsamlı bir şekilde ele alabilecek şekilde belirlenmelidir. Örneğin bir e-ticaret veritabanı projesi için "Müşteri istediği ürünleri sepete ekleyip satın alabilecek" gereksinimi müşteri rolündeki kullanıcı için bir gereksinim örneğidir, "Yöneticiler, ürün bilgilerini düzenleyebilecek veya yayından kaldırabilecektir" yönetici rolündeki kullanıcı için bir gereksinim örneğidir.

İlk önce E-R diyagramını aşağıdaki gibi oluşturduk. Daha sonrasında E-R diyagramımızdaki varlıkların ve niteliklerin veri tabanı tablolalarını oluşturduk. Tablo oluşturma SQL komutlarımız aşağıdaki gibidir.

```
--PROJE GEREKSİNİMLERİNDE GERÇEKLEŞTİRİLECEK İŞLEMLER İÇİN GEREKLİ SQL KOMUTLARI
-- Personel tablosu
CREATE TABLE Personel (
    PersonelID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(100) NOT NULL,
    Soyad NVARCHAR(100) NOT NULL,
    Telefon NVARCHAR(15) UNIQUE NOT NULL, -- Telefon numaraları benzersiz ve boş olamaz
    YöneticiID INT NULL,
    FOREIGN KEY (YöneticiID) REFERENCES Personel(PersonelID)
);

-- Müşteri tablosu
CREATE TABLE Müşteri (
    MüşteriID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(100) NOT NULL,
    Soyad NVARCHAR(100) NOT NULL,
    Telefon NVARCHAR(15) UNIQUE NOT NULL, -- Telefon numaraları benzersiz ve boş olamaz
    Email NVARCHAR(100) UNIQUE NOT NULL -- Email adresleri benzersiz ve boş olamaz
);

CREATE TABLE Yorum (
    YorumID INT PRIMARY KEY IDENTITY(1,1),
    MüşteriID INT, -- Yorum yapan müşteri
    OtoparkID INT, -- Yorum yapılan otopark
    YorumMetni NVARCHAR(500), -- Yorum metni
    Puan INT CHECK (Puan >= 1 AND Puan <= 5), -- 1-5 arasında puan
    YorumTarihi DATETIME DEFAULT GETDATE(), -- Yorum tarihi
    FOREIGN KEY (MüşteriID) REFERENCES Müşteri(MüşteriID)

);
-- Otopark ve Yorum arasındaki M:N ilişkiyi kuran tablonun oluşturulması
CREATE TABLE OtoparkYorum (
    OtoparkID INT NOT NULL,
    YorumID INT NOT NULL,
    PRIMARY KEY (OtoparkID, YorumID),
    FOREIGN KEY (OtoparkID) REFERENCES Otopark(OtoparkID),
    FOREIGN KEY (YorumID) REFERENCES Yorum(YorumID)
);

-- Araç tablosu
CREATE TABLE Araç (
    AraçID INT PRIMARY KEY IDENTITY(1,1),
    MüşteriID INT NOT NULL,
    PlakaNo NVARCHAR(15) UNIQUE NOT NULL, -- Plaka numaraları benzersiz ve boş olamaz
    Marka NVARCHAR(50),
    Model NVARCHAR(50),

    FOREIGN KEY (MüşteriID) REFERENCES Müşteri(MüşteriID)
);
```

```
-- Otopark tablosu
CREATE TABLE Otopark (
    OtoparkID INT PRIMARY KEY IDENTITY(1,1),
    ParkYeriNo INT UNIQUE NOT NULL, -- Her otopark alan numarası benzersiz ve boş olamaz
    Kapasite INT NOT NULL,
    Konum NVARCHAR(255) NOT NULL,
    Durum NVARCHAR(25)
);

-- Abonelik tablosu
CREATE TABLE Abonelik (
    AbonelikID INT PRIMARY KEY IDENTITY(1,1),
    MüşteriID INT NOT NULL,
    BaşlangıçTarihi DATE NOT NULL,
    BitişTarihi DATE NOT NULL,
    Durum NVARCHAR(50) ,
    FOREIGN KEY (MüşteriID) REFERENCES Müşteri(MüşteriID),
    CHECK (BitişTarihi >= BaşlangıçTarihi) -- Tarih kontrolü
);

-- Ödeme tablosu
CREATE TABLE Ödeme (
    ÖdemeID INT PRIMARY KEY IDENTITY(1,1),
    MüşteriID INT NOT NULL,
    PersonelID INT ,
    ÖdemeTarihi DATE NOT NULL,
    Tutar DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (MüşteriID) REFERENCES Müşteri(MüşteriID),
    FOREIGN KEY (PersonelID) REFERENCES Personel(PersonelID)

);

ALTER TABLE Araç
ADD ParkYeriNo INT,
    GirişTarihSaat DATETIME,
    ÇıkışTarihSaat DATETIME,
    FOREIGN KEY (ParkYeriNo) REFERENCES Otopark(ParkYeriNo);


ALTER TABLE Otopark
ADD CONSTRAINT UQ_Otopark_Konum UNIQUE (Konum);


ALTER TABLE Araç
ADD Konum NVARCHAR(255),
    FOREIGN KEY (Konum) REFERENCES Otopark(Konum);
```

İlişki Türlerinin Özeti
:
1:N İlişkisi:

Personel → Personel (Yönetici-Alt çalışan)
Müşteri → Yorum
Müşteri → Araç
Müşteri → Abonelik
Müşteri → Ödeme
Personel → Ödeme
Otopark → Araç (Park yeri)

N:1 İlişkisi:

Araç → Otopark (Her araç sadece bir park yerinde olabilir, ancak bir park yerinde birden fazla araç olabilir)

N:M İlişkisi:
Otopark → Yorum (Bu yüzden ayrı bir ilişki tablosu yaptık.)

--PROJE GEREKSİNİMLERİ GÖSTERMEK İÇİN ÖRNEK VERİLER EKLEME KOMUTLARI

```
-- 1. Personel Tablosuna Veri Ekleme
INSERT INTO Personel (Ad, Soyad, Telefon, YöneticiID)
VALUES
('Ahmet', 'Yılmaz', '5551234567', NULL),  -- Yönetici olmayan bir personel
('Mehmet', 'Kaya', '5552345678', 1),      -- Yönetici olarak Ahmet'in altındaki personel
('Ayşe', 'Çelik', '5553456789', 1),       -- Yönetici olarak Ahmet'in altındaki personel
('Fatma', 'Demir', '5554567890', 2);      -- Yönetici olarak Mehmet'in altındaki personel

-- 2. Müşteri Tablosuna Veri Ekleme
INSERT INTO Müşteri (Ad, Soyad, Telefon, Email)
VALUES
('Ali', 'Veli', '5555678901', 'ali.veli@example.com'),
('Zeynep', 'Güler', '5556789012', 'zeynep.guler@example.com'),
('Okan', 'Kara', '5557890123', 'okan.kara@example.com'),
('Merve', 'Şahin', '5558901234', 'merve.sahin@example.com');

-- 3. Otopark Tablosuna Veri Ekleme
INSERT INTO Otopark (ParkYeriNo, Kapasite, Konum, Durum)
VALUES
(1, 50, 'Merkez Mahallesi', 'Dolu'),
(2, 100, 'Doğu Sanayi', 'Boş'),
(3, 75, 'Batı Limanı', 'Dolu'),
(4, 120, 'Güney Çarşı', 'Boş');

-- 4. Araç Tablosuna Veri Ekleme
INSERT INTO Araç (MüşteriID, PlakaNo, Marka, Model, ParkYeriNo, GirişTarihSaat, ÇıkışTarihSaat)
VALUES
(1, '34ABC123', 'Ford', 'Fiesta', 1, '2025-01-01 08:00:00', '2025-01-01 12:00:00'),
(2, '35DEF456', 'Toyota', 'Corolla', 2, '2025-01-02 09:30:00', '2025-01-02 11:30:00'),
(3, '36GHI789', 'BMW', '3 Series', 3, '2025-01-03 10:15:00', '2025-01-03 14:15:00'),
(4, '37JKL012', 'Audi', 'A4', 4, '2025-01-04 07:00:00', '2025-01-04 09:00:00');

-- 5. Yorum Tablosuna Veri Ekleme
INSERT INTO Yorum (MüşteriID, OtoparkID, YorumMetni, Puan)
VALUES
(1, 1, 'Otopark çok kalabalıktı, ama hizmet kalitesi iyi.', 4),
(2, 2, 'Park alanı geniş ve rahat, çok memnun kaldım.', 5),
(3, 3, 'Otopark biraz zor bulunuyor, ama genel olarak iyi.', 3),
(4, 4, 'Park etmek zor, fakat güvenlik önlemleri oldukça iyi.', 4);

-- 6. Abonelik Tablosuna Veri Ekleme
INSERT INTO Abonelik (MüşteriID, BaşlangıçTarihi, BitişTarihi, Durum)
VALUES
(1, '2025-01-01', '2025-06-01', 'Aktif'),
(2, '2025-02-15', '2025-08-15', 'Aktif'),
(3, '2025-03-01', '2025-09-01', 'Pasif'),
(4, '2025-04-01', '2025-10-01', 'Aktif');

-- 7. Ödeme Tablosuna Veri Ekleme
INSERT INTO Ödeme (MüşteriID, PersonelID, ÖdemeTarihi, Tutar)
VALUES
(1, 2, '2025-01-01', 150.75),
(2, 3, '2025-02-15', 200.50),
(3, 1, '2025-03-10', 180.00),
(4, 2, '2025-04-01', 210.25);
```

YAPTIĞIMIZ STORE PROCEDURE ÖRNEKLERİ 1)

--TEK BİR SAKLI YORDAM ÇAĞRISI İLE HEM MÜŞTERİ HEM DE ARAÇ KAYDINI BİRARADA YAPMAK İÇİN YAZDIK

```
CREATE PROCEDURE InsertMusteriVeArac
    @Ad NVARCHAR(100),
    @Soyad NVARCHAR(100),
    @Telefon NVARCHAR(15),
    @Email NVARCHAR(100),
    @PlakaNo NVARCHAR(15),
    @Marka NVARCHAR(50),
    @Model NVARCHAR(50),
    @ParkYeriNo INT,
    @Konum NVARCHAR(100),
    @GirişTarihSaat DATETIME -- Yeni parametre
AS
BEGIN
    -- Müşteri tablosuna yeni müşteri ekleyelim
    INSERT INTO Müşteri (Ad, Soyad, Telefon, Email)
    VALUES (@Ad, @Soyad, @Telefon, @Email);

    -- Yeni eklenen müşterinin MüşteriID'sini alalım
    DECLARE @MüşteriID INT;
    SET @MüşteriID = SCOPE_IDENTITY();

    -- Araç tablosuna yeni araç kaydedelim
    INSERT INTO Araç (MüşteriID, PlakaNo, Marka, Model, ParkYeriNo, Konum, GirişTarihSaat)
    VALUES (@MüşteriID, @PlakaNo, @Marka, @Model, @ParkYeriNo, @Konum, @GirişTarihSaat);
END;

--ÖRNEĞİ
EXEC InsertMusteriVeArac
    @Ad = 'Ahmet',
    @Soyad = 'Yılmaz',
    @Telefon = '05551234567',
    @Email = 'ahmet@example.com',
    @PlakaNo = '34AVD123',
    @Marka = 'Ford',
    @Model = 'Focus',
    @ParkYeriNo = 14,
    @Konum = 'Şehir Merkezi',
    @GirisTarihi = '2025-01-04 08:30:00';
```

YAPTIĞIMIZ STORE PROCEDURE ÖRNEKLERİ 2)

```
-- Tek bir saklı yordam çağrısıyla:
--1.        Yeni bir müşteri kaydı eklemek.
--2.        Bu müşteriye bir abonelik oluşturmak
CREATE OR ALTER PROCEDURE AddCustomerAndSubscription
    @Ad NVARCHAR(100),
    @Soyad NVARCHAR(100),
    @Telefon NVARCHAR(15),
    @Email NVARCHAR(100),
    @BaşlangıçTarihi DATE,
    @BitişTarihi DATE
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        -- Müşteri ekle
        INSERT INTO Müşteri (Ad, Soyad, Telefon, Email)
        VALUES (@Ad, @Soyad, @Telefon, @Email);

        -- Yeni eklenen müşterinin ID'sini al
        DECLARE @MüşteriID INT = SCOPE_IDENTITY();

        -- Abonelik ekle
```

```
        INSERT INTO Abonelik (MüşteriID, BaşlangıçTarihi, BitişTarihi, Durum)
        VALUES (@MüşteriID, @BaşlangıçTarihi, @BitişTarihi, NULL);

        COMMIT TRANSACTION;        -- Eğer her şey başarılıysa, işlemi tamamla
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;  -- Hata durumunda işlemi geri al
        THROW;                 -- Hata mesajını kullanıcıya ilet
    END CATCH;
END;
GO
--ÖRNEĞİ
EXEC InsertMusteriVeArac
    @Ad = 'Ahmet',
    @Soyad = 'Yılmaz',
    @Telefon = '05551234567',
    @Email = 'ahmet@example.com',
    @PlakaNo = '34AVD123',
    @Marka = 'Ford',
    @Model = 'Focus',
    @ParkYeriNo = 14,
    @Konum = 'Şehir Merkezi',
    @GirisTarihi = '2025-01-04 08:30:00';
--BU KODDA TRANSACTİON DA BULUNUYOR (COMMİT VE ROLLBACK) ÖRNEK TEST SENARYOSUDA ŞÖYLE
--Hata vermesi gereken kod
EXEC AddCustomerAndSubscription
    @Ad = 'Ahmet',
    @Soyad = 'Yılmaz',
    @Telefon = '5555555555',
    @Email = 'ali.ozturk@example.com',  -- Bu email zaten mevcutsa hata verecek
    @BaşlangıçTarihi = '2025-01-01',
    @BitişTarihi = '2025-12-31';
--HATA KODU
--(0 rows affected)
--Msg 2627, Level 14, State 1, Procedure AddCustomerAndSubscription, Line 13 [Batch Start Line 0]
--Violation of UNIQUE KEY constraint 'UQ_Müşteri_A9D105345945F31F'. Cannot insert duplicate key in object
'dbo.Müşteri'. The duplicate key value is (ali.ozturk@example.com).
--Completion time: 2025-01-04T23:29:36.7053085+03:00


YAPTIĞIMIZ TRİGGER ÖRNEKLERİ 1)

--VERİ EKLENDİKTEN SONRA OTOPARK DURUMU BOŞ OLAN YERİ DOLU OLARAK GÜNCELLEYEN TRİGGER
CREATE TRIGGER UpdateOtoparkDurum
ON Araç
AFTER INSERT
AS
BEGIN
    -- Araç tablosuna yeni bir kayıt eklenmişse
    -- Otopark tablosunda, ParkYeriNo ve Konum'a göre Durum'u 'Dolu' olarak güncelle
    UPDATE Otopark
    SET Durum = 'Dolu'
    FROM Otopark o
    INNER JOIN inserted i
        ON o.ParkYeriNo = i.ParkYeriNo
        AND o.Konum = i.Konum;
END;

--ÖRNEĞİ
INSERT INTO Araç (MüşteriID, PlakaNo, Marka, Model, ParkYeriNo, Konum, GirişTarihiSaat)
VALUES (10, '65AD123', 'BMW', '320d', 11, 'Şehir Merkezi', '2025-01-04 14:30:00');
-- Trigger 1
```

YAPTIĞIMIZ TRİGGER ÖRNEKLERİ 2)

--Eğer BitişTarihi şu anki tarihten geçmişse Durum 'Pasif' olarak güncellenir. Eğer bitiş tarihi henüz gelmemişse Durum 'Aktif' olur.(TRİGGER)

```
CREATE TRIGGER UpdateAbonelikDurum
ON Abonelik
AFTER INSERT, UPDATE
AS
BEGIN
   -- Durum güncelleme işlemi
   DECLARE @BitişTarihi DATE;
   DECLARE @Durum NVARCHAR(50);

   -- Insert edilen yeni aboneliğin bitiş tarihini al
   SELECT @BitişTarihi = BitişTarihi FROM INSERTED;

   -- Eğer bitiş tarihi geçmişse Durum 'Pasif', geçmiş değilse 'Aktif' olur
   IF @BitişTarihi < GETDATE()
   BEGIN
      SET @Durum = 'Pasif';
   END
   ELSE
   BEGIN
      SET @Durum = 'Aktif';
   END

   -- Durumu güncelle
   UPDATE Abonelik
   SET Durum = @Durum
   WHERE BitişTarihi = @BitişTarihi;
END;
--ÖRNEĞİ
-- Abone 1: Bitiş tarihi geçmiş (Durum = Pasif olacak)
INSERT INTO Abonelik (MüşteriID, BaşlangıçTarihi, BitişTarihi, Durum)
VALUES (1, '2024-01-01', '2024-12-31', NULL);

-- Abone 2: Bitiş tarihi şu anki tarih (Durum = Aktif olacak)
INSERT INTO Abonelik (MüşteriID, BaşlangıçTarihi, BitişTarihi, Durum)
VALUES (2, '2024-01-01', '2025-05-05', NULL);

-- Abone 3: Bitiş tarihi geçmiş (Durum = Aktif olacak)
INSERT INTO Abonelik (MüşteriID, BaşlangıçTarihi, BitişTarihi, Durum)
VALUES (3, '2023-01-01', '2026-05-01', NULL);

-- Abone 4: Bitiş tarihi geçmiş (Durum = Pasif olacak)
INSERT INTO Abonelik (MüşteriID, BaşlangıçTarihi, BitişTarihi, Durum)
VALUES (4, '2023-06-01', '2024-01-01', NULL);
```

İşlevsel Gereksinimler

Kullanıcı Yönetimi:
Müşteri kayıtlarının yapılması.
Personel kayıtlarının yapılması ve yönetici-personel ilişkilerinin tanımlanması.

Otopark Yönetimi:
Otoparkların tanımlanması (konum, kapasite, durum bilgileri).
Her otopark için farklı park yerlerinin tanımlanması ve durumlarının güncellenmesi (dolu, boS)

Araç Yönetimi:
Müşterilerin araçlarının kaydedilmesi (plaka, marka, model bilgileri).
Araçların giriş ve çıkış saatlerinin kaydedilmesi.
Park yeri atamalarının yapılması.

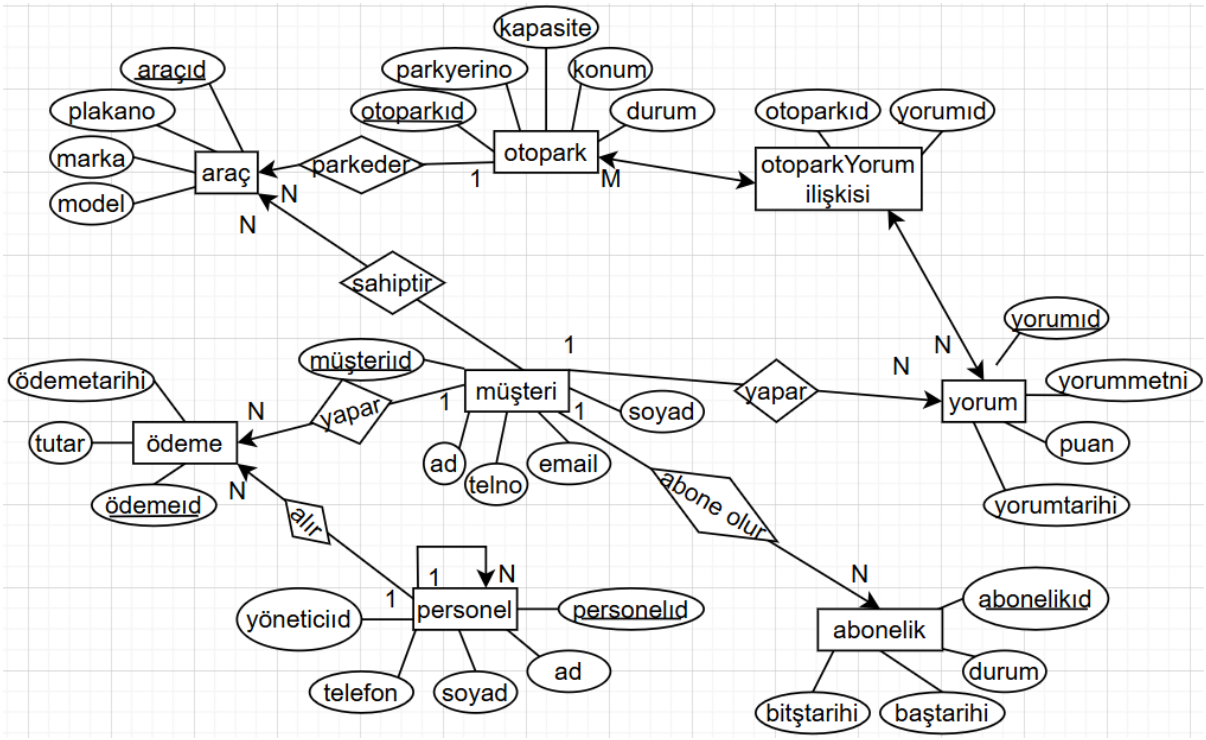
Yorum Yönetimi:
Müşterilerin otoparklar hakkında yorum yapabilmesi.
Otoparklara yapılan yorumların listelenmesi ve yönetilmesi.
Yorumların puan aralıklarına göre filtrelenmesi.

Abonelik Yönetimi:
Müşteriler için abonelik tanımlanması (başlangıç ve bitiş tarihleri, durum bilgisi).
Aktif ve pasif aboneliklerin görüntülenmesi.

Ödeme Yönetimi:
Müşterilerin ödemelerinin kaydedilmesi.
Her ödemenin tutarı, tarihi ve ödeme yapan müşteri/personel bilgilerinin kaydedilmesi.

2- E-R Diyagramı

Proje gereksinimlerine göre E-R diyagramı oluşturulacaktır.


kapasite
araçid
parkyerino
konum
plakano
otoparkid
durum
otoparkid
yorumid
marka
araç
parkeder
otopark
otoparkYorum
ilişkisi
model
1
M
N
N
sahiptir
1
yorumid
müşteriid
N
N
ödemetarihi
müşteri
yapar
yapar
yorummetni
N
1
1
soyad
tutar
ödeme
yorum
puan
ad
email
telno
N
ödemeid
yorumtarihi
alır
abone olur
1
N
N
1
yöneticiid
personel
personelid
abonelik
abonelikid
ad
durum
telefon
soyad
bitştarihi
baştarihi