T.C.

FIRAT ÜNİVERSTESİ

MÜHENDİSLİK FAKÜLTESİ

KARATE KLÜBÜ

HAZIRLAYANLAR

MUSTAFA RIDVAN

BEŞŞAR ELHATİB

KADİR BERKE KÖKSAL

DERS HOCALARI

Dr.Öğr.Üyesi ERTAN BÜTÜN

Arş.Gör. MUSA YENİLMEZ

Biilgisayar Mühendisliği

Veri Taban Projesi

Ocak 2025

ELAZIĞ

İçindekiler :

Karate Kulübü için Veritabanı Gereksiniml

Kemerler Tablosu:
Karate kulübümüzde toplam 8 kemer bulunmaktadır ve her kemerin bir rengi vardır. Kemer renkleri şu şekildedir: Siyah, Kahverengi, Mor, Mavi, Yeşil, Turuncu, Sarı ve Beyaz. En üst seviye kemer Siyah kemerdir.

Kisiler Tablosu:
Bu tablo, kulüpte bulunan tüm kişilerin (hem üyelerin hem de eğitmenlerin) bilgilerini içerir. Her kişi için aşağıdaki bilgiler yer alır:

- Adı
- Soyadı
- Telefon numarası
- E-posta adresi
- Adres

Egitmenler Tablosu:
Eğitmenler tablosu, sadece eğitmenlere ait bilgileri içerir. Her eğitmen, Kisiler tablosundaki bir kişinin numarasını (kisiId) kullanır. Yani bir kişi eğitmen olduğu zaman, onun bilgileri bu tabloya taşınır.

Uyeler Tablosu:
Üyeler tablosu, sadece kulüp üyelerine ait bilgileri içerir. Her üye de Kisiler tablosundan bir numara (kisiId) alır.
Bir kişi, eğitmen olduğunda üye olamaz. Aynı şekilde bir üye, eğitmen olarak atanamaz.

Odemeler Tablosu:
Bu tablo, tüm ödeme kayıtlarını tutar ve şu bilgileri içerir:

- Ödeme numarası (oId)
- Ödeme tarihi (odemeTarihi)
- Ödenen miktar (paraMiktari)

- Kulüp kasasında bulunan toplam bakiye (genelBakiye)

EgitmenKontrati Tablosu:

Eğitmenlere sözleşme yapmak için ödeme yapılması gereklidir. Bu nedenle, EgitmenKontrati tablosu, bir eğitmenin kontrat bilgilerini içerir ve aşağıdaki alanlara sahiptir:
Eğitmen numarası (egitmenId)

- Ödeme numarası (oId)
- Sözleşmenin başlangıç tarihi (kontratBaslangicTarihi)
- Sözleşmenin bitiş tarihi (kontratBitisTarihi)

Uyelik Tablosu:

Bir üyenin aktif hale gelebilmesi için ödeme yapması gereklidir. Bu nedenle, üyelik bilgileri Uyelik tablosunda saklanır.

Antrenman Tablosu:

Bu tablo, bir üyenin bir eğitmenle yaptığı antrenmanları içerir. Aşağıdaki bilgiler bulunur:

- Eğitmen numarası (eId)
- Üye numarası (kisiId)
- Antrenman tarihi (atmaTarihi)

Yarismalar Tablosu:

Yarışmalarla ilgili bilgiler bu tabloda yer alır ve şunları içerir:

- Yarışma numarası (yId)
- Yarışma adı (yarismaAdi)
- Yarışma tarihi (yarismaTarihi)

YarismaKatilim Tablosu:

Bu tablo, üyelerin yarışmalara katılımını kaydeder ve şu alanlara sahiptir:

- Yarışma numarası (yId)
- Üye numarası (kisiId)
- Kazandı mı (kazandi - 1 ise kazandı, 0 ise kazanmadı)

Odulleri Tablosu:

Eğer bir üye bir yarışmayı kazanırsa, bu tablonun içine bir ödül kaydı eklenir. Tablo şu bilgileri içerir:

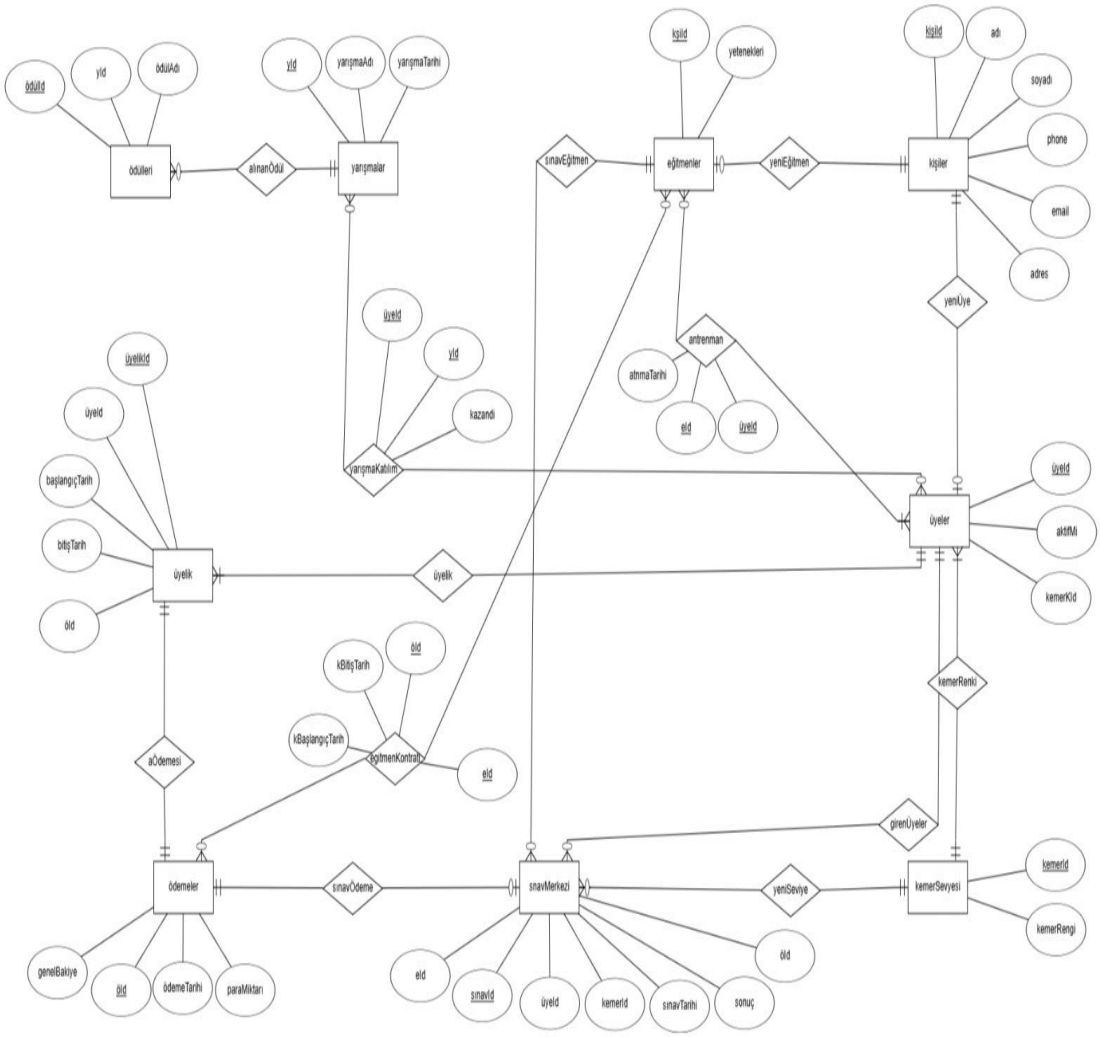
- Ödül numarası (odulId)
- Yarışma numarası (yId)
- Ödül adı (odulAdi)

SinavMerkezi Tablosu:
Sınav Merkezi tablosu, karate kulübündeki sınavlarla ilgili bilgileri saklar. Bu sınavlar, üyelerin kemer seviyesini artırmak veya mevcut seviyelerini değerlendirmek için yapılır. Tablo şu bilgileri içerir:

- Sınav Numarası (sinavId): Her sınav için benzersiz bir kimlik numarası.
- Kişi Numarası (kisiId): Sınava giren üyenin numarası.
- Kemer Numarası (kemerId): Sınavda hangi kemer seviyesi için test yapıldığını belirtir.
- Ödeme Numarası (oId): Sınav ücreti ödemesinin kaydedildiği ödeme numarası.
- Eğitmen Numarası (eId): Sınavı gerçekleştiren eğitmenin numarası.
- Sınav Tarihi (sinavTarihi): Sınavın yapıldığı tarih.
- Sonuç (sonuc): Üyenin sınavdaki başarısını gösterir. Örneğin, "1" başarılı, "0" başarısız anlamına gelebilir.

Sınav Merkezi tablosu, sınavın sonuçlarına göre üyelerin kemer seviyelerinin güncellenmesini sağlar. Bu süreç aşağıdaki kurallara göre yürütülür:
Başarılı Olan Üyeler (sonuc = "1"):
Eğer öğrenci sınavda başarılı olursa (sonuc değeri "1" ise), üyenin kemer seviyesi bir üst seviyeye yükseltilir.
Bu güncelleme, KemerSeviyesi tablosu ile bağlantılı olarak gerçekleştirilir. Üyenin mevcut kemerId değeri, bir üst seviyedeki kemerId ile değiştirilir.
Kemer Seviyesi Siyah Olan Üyeler:
Siyah kemer (en yüksek seviye) sahibi olan üyeler sınava giremez.
Bu, sistemde bir kontrol mekanizması ile sağlanır. Eğer bir üyenin kemerId değeri siyah kemerle eşleşiyorsa, sınav kaydı oluşturulmasına izin verilmez.

E-R DYAGRAM


ödülId
yId
ödülAdı
ödülleri
alınanÖdül
yId
yarışmaAdı
yarışmaTarihi
yarışmalar
kişiId
yetenekleri
sınavEğitmen
eğitmenler
yeniEğitmen
kişiId
adı
soyadı
phone
email
adres
kişiler
yeniÜye
üyeId
yId
kazandı
yarışmaKatılım
antrenman
alınmaTarihi
eId
üyeId
üyelikId
üyeId
başlangıçTarih
bitişTarih
üyelik
üId
üyelik
üyeId
aktifMi
kemerKId
üyeler
kBitişTarih
üId
kBaşlangıçTarih
eğitmenKontrat
eId
kemerRenki
aÖdemesi
girenÜyeler
ödemeler
sınavÖdeme
sınavMerkezi
yeniSeviye
kemerSeviyesi
kemerId
kemerRengi
genelBakiye
üId
ödemeTarihi
paraMiktarı
eId
sınavId
üyeId
kemerId
sınavTarihi
sonuç
üId

Tabloları:

- **Kisiler**:(kisiId, adı, soyadı, phone, email, adres)
- **Egitmenler**: (kisiId ,yetenekleri)
- **Uyeler**: (kisiId, aktifMi , kemerId)
- **Uyelik**:(uyelikId, uyeId, oId ,baslangicTarih, bitisTarih)
- **EgitmenKontrati:**(eId, oId,,uyeId,kBitişTarih,kBaslangicTarihi)
- **Antrenman**: (eId, UyeId , atmaTarihi)
- **KemerSeviyesi**: (kemerId, kemerRengi)
- **SinavMerkezi**: (sınavId, eId ,uyeId , kemerId ,oId ,sinavTarihi, sonuc)
- **Yarısmalar**: (yId, yarısmaAdi, yarısmaTarihi)
- **YarısmaKatılım :** (yId, uyeId,kazandi)
- **Odulleri** : (odulId, yId, odulAdi)
- **Odemeler**: (oId, odemeTarihi, paraMiktari,genelBakiye)

İlişkiler:

- **Kişiler - Üyeler**: 1:0..1 (Her kişi ya üye ya da eğitmen olabilir.)
- **Kişiler - Eğitmenler**: 1:0..1 (Her kişi ya eğitmen ya da üye olabilir.)
- **Eğitmenler - Üyeler**: M:M (Her eğitmen, birden fazla üyeye ders verebilir ve her üye, birden fazla eğitmenden ders alabilir.)
- **Eğitmenler - Sınav Merkezleri**: 1:N (Her sınav merkezi, birden fazla eğitmeni sınavları yönetmekle görevlendirebilir.)
- **Üyeler - Kemer Seviyesi**: M:1 (Her üye, bir kemer seviyesine sahip olmalıdır.)
- **Üyeler - Sınav Merkezleri**: 1:0..M (Her üye, sınav merkezlerinde sınavlara katılabilir veya katılmayabilir.)
- **Üyeler - Üyelik**: 1:M (Her üye, birden fazla üyelik oluşturabilir.)
- **Üyeler - Üye Kartı**: 1:1 (Her üyenin mutlaka bir üye kartı olmalıdır.)
- **Üyeler - Yarışmalar**: 0..N:0..M (Her üye, birden fazla yarışmaya katılabilir veya katılmayabilir.)
- **Yarışmalar - Ödüller**: 1:0..N (Her yarışma, birden fazla ödül kazanabilir ya da ödül almayabilir.)
- **Üyelik - Ödemeler**: 1:1 (Her üyelik için bir ödeme yapılması gerekmektedir.)
- **Eğitmenler - Eğitmen Kontratı**: 1:M (Her eğitmen, birden fazla kontrata sahip olabilir, her kontrat bir başlangıç ve bitiş tarihine sahip olacaktır.)
- **Eğitmen Kontratı - ödemeler**: M:1
- **Sınav Merkezleri - Eğitmen** : 1:N (Her eğitmen birden fazla sınaf sorumlusu olabilir)
- **Sınav Merkezleri - Üyeler**: 1:M (Her üye, birden fazla sınav merkezinde sınavlara katılabilir.)
- **Sınav Merkezleri – KamerSevyesi**: 1:N

Karate Kulübü Veritabanı Tasarımı: BCNF Uyumunun

Bir veritabanı tasarımının tam normal form (BCNF) uyumlu olduğunu söyleyebilmek için her tablonun işlevsel bağımlılıkları analiz edilmeli ve tabloların şu kuralları sağlaması gerekmektedir:

- Her işlevsel bağımlılık (functional dependency), yalnızca tabloların birincil anahtarına bağlı olmalıdır.
- Kısmi bağımlılık (partial dependency) veya transitif bağımlılık (transitive dependency) bulunmamalıdır.

Karate kulübü veritabanındaki tüm tablolar bu kuralları sağlamaktadır. Aşağıda bu analiz ayrıntılı bir şekilde sunulmuştur:

---

1. Kisiler Tablosu

- **Birincil Anahtar:** kisiId
- **Diğer Alanlar:** adi, soyadi, telefon, eposta, adres

Analiz:

- Her bir alan, doğrudan kisiId birincil anahtarına bağımlıdır.
- Örneğin, bir kişinin adı (adi) yalnızca o kişinin benzersiz kisiId değeriyle ilişkilidir.
- Kısmi veya transitif bağımlılık bulunmamaktadır.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

2. Egitmenler Tablosu

- **Birincil Anahtar:** kisiId (aynı zamanda yabancı anahtar olarak Kisiler tablosuna bağlıdır).
- **Diğer Alanlar:** yetenekleri

Analiz:

- yetenekleri alanı yalnızca kisiId anahtarına bağımlıdır.
- kisiId, her eğitmeni benzersiz bir şekilde tanımlar ve başka bağımlılık yoktur.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

3. KemerSeviyesi Tablosu

- **Birincil Anahtar:** kemerId
- **Diğer Alanlar:** kemerRengi

Analiz:

- kemerRengi alanı, yalnızca kemerId anahtarına bağımlıdır.
- Her kemerId değeri benzersiz bir kemer rengine karşılık gelir.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

4. Uyeler Tablosu

- **Birincil Anahtar:** kisiId (aynı zamanda yabancı anahtar olarak Kisiler tablosuna bağlıdır).
- **Diğer Alanlar:** aktifMi, kemerId

Analiz:

- aktifMi ve kemerId alanları, yalnızca kisiId anahtarına bağımlıdır.
- kemerId, KemerSeviyesi tablosuna bağlıdır, ancak bu bağlanma doğrudan kisiId üzerinden gerçekleştirilir. Transitif bağımlılık yoktur.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

5. Odemeler Tablosu

- **Birincil Anahtar:** oId
- **Diğer Alanlar:** odemeTarihi, paraMiktari, genelBakiye

Analiz:

- Tüm alanlar yalnızca oId anahtarına bağımlıdır. Örneğin, bir ödeme tarihinin (odemeTarihi) yalnızca ilgili oId ile ilişkilendirildiği açıktır.
- Kısmi veya transitif bağımlılık bulunmamaktadır.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

6. Uyelik Tablosu

- **Birincil Anahtar:** uyelikId
- **Diğer Alanlar:** kisiId, oId, baslangicTarih, bitisTarih

Analiz:

- baslangicTarih ve bitisTarih gibi alanlar yalnızca uyelikId anahtarına bağımlıdır.
- kisiId ve oId alanları, birincil anahtar (uyelikId) üzerinden birbirine bağlanır. Bu, transitif bağımlılık olmadığını gösterir.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

7. Antrenman Tablosu

- **Birincil Anahtar:** (kisiId, eId) (bileşik anahtar)
- **Diğer Alanlar:** atmaTarihi

Analiz:

- atmaTarihi alanı, yalnızca (kisiId, eId) bileşik anahtarına bağımlıdır.
- Örneğin, bir kişinin belirli bir eğitmenle yaptığı antrenman tarihi, yalnızca bu iki değer üzerinden tanımlanabilir.
- Kısmi bağımlılık yoktur; her alan, bileşik anahtarın tamamına bağımlıdır.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

8. SinavMerkezi Tablosu

- **Birincil Anahtar:** sinavId
- **Diğer Alanlar:** kisiId, kemerId, oId, eId, sinavTarihi, sonuc

Analiz:

- Tüm alanlar, yalnızca sinavId birincil anahtarına bağımlıdır.
- Örneğin, bir sınav sonucu (sonuc) yalnızca ilgili sinavId ile ilişkilidir. Başka bir transitif veya kısmi bağımlılık yoktur.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

9. Yarismalar Tablosu

- **Birincil Anahtar:** yId
- **Diğer Alanlar:** yarismaAdi, yarismaTarihi

Analiz:

- yarismaAdi ve yarismaTarihi alanları yalnızca yId anahtarına bağımlıdır.
- Her yarışma benzersiz bir kimlik (yId) ile tanımlandığından, kısmi veya transitif bağımlılık bulunmamaktadır.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

10. YarismaKatilim Tablosu

- **Birincil Anahtar:** (yId, kisiId) (bileşik anahtar)
- **Diğer Alanlar:** kazandi

Analiz:

- kazandi alanı yalnızca (yId, kisiId) bileşik anahtarına bağımlıdır.
- Bu, her katılımın sonuçlarını tanımlamak için gereklidir. Kısmi bağımlılık bulunmamaktadır.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

11. Odulleri Tablosu

- **Birincil Anahtar:** odulId
- **Diğer Alanlar:** yId, odulAdi

Analiz:

- odulAdi alanı yalnızca odulId anahtarına bağımlıdır.
- yId, ödülün hangi yarışmaya ait olduğunu belirtir, ancak bu ilişki doğrudan anahtar üzerinden tanımlanır.

Sonuç: Bu tablo BCNF'ye uyumludur.

---

12. EgitmenKontrati Tablosu

- **Birincil Anahtar:** (egitmenId, oId) (bileşik anahtar)
- **Diğer Alanlar:** kontratBaslangicTarihi, kontratBitisTarihi

Analiz:

- kontratBaslangicTarihi ve kontratBitisTarihi alanları yalnızca (egitmenId, oId) bileşik anahtarına bağımlıdır.
- egitmenId, sözleşmenin hangi eğitmene ait olduğunu belirtirken, oId, sözleşmenin ilgili ödeme kaydıyla ilişkilendirildiğini ifade eder.

- Tüm alanlar doğrudan bileşik anahtar (egitmenId, oId) üzerinden tanımlanır; başka bir transitif veya kısmi bağımlılık bulunmamaktadır.

- **Sonuç:** Bu tablo BCNF'ye uyumludur.

Genel Sonuç: Bu veritabanı tasarımındaki tüm tablolar, BCNF'nin gerekliliklerini karşılamaktadır. Tüm işlevsel bağımlılıklar yalnızca birincil anahtarlara bağlıdır, ve kısmi veya transitif bağımlılık bulunmamaktadır. Tasarım, veri tekrarı ve tutarsızlıkları önlemek için optimize edilmiştir.

SQL KOMUTLARI

Veri Taban Ve Tablolar Oluşturma

```
CREATE DATABASE karateKlubu;
USE karateKlubu;

-- Kişiler tablosu
CREATE TABLE Kisiler (
    kisiId INT PRIMARY KEY IDENTITY(1,1),
    adi NVARCHAR(50),
    soyadi NVARCHAR(50),
    telefon NVARCHAR(20),
    eposta NVARCHAR(50),
    adres NVARCHAR(255)
);

-- Ödemeler tablosu
CREATE TABLE Odemeler (
    oId INT PRIMARY KEY IDENTITY(1,1),
    odemeTarihi DATE,
    paraMiktari FLOAT,
    genelBakiye DECIMAL(15,2)
);

-- Eğitmenler tablosu
CREATE TABLE Egitmenler (
    kisiId INT PRIMARY KEY,
    yetenekleri NVARCHAR(255),
    CONSTRAINT fk_egitmenler_kisiler FOREIGN KEY (kisiId) REFERENCES
Kisiler(kisiId)
);

-- Kemer Seviyesi tablosu
CREATE TABLE KemerSeviyesi (
    kemerId INT PRIMARY KEY IDENTITY(1,1),
    kemerRengi NVARCHAR(50)
);

-- Üyeler tablosu
CREATE TABLE Uyeler (
```

```
    kisiId INT PRIMARY KEY,
    aktifMi BIT,
    kemerId INT,
    CONSTRAINT fk_uyeler_kisiler FOREIGN KEY (kisiId) REFERENCES
Kisiler(kisiId),
    CONSTRAINT fk_uyeler_kemer FOREIGN KEY (kemerId) REFERENCES
KemerSeviyesi(kemerId)
);

-- Eğitmen Kontratı tablosu
CREATE TABLE EgitmenKontrati (
    egitmenId INT NOT NULL,
    oId INT NOT NULL,
    kontratBaslangicTarihi DATE NULL,
    kontratBitisTarihi DATE NOT NULL,
    PRIMARY KEY (egitmenId, oId),
    CONSTRAINT fk_egitmenkontrati_egitmenler FOREIGN KEY (egitmenId)
REFERENCES Egitmenler(kisiId),
    CONSTRAINT fk_egitmenkontrati_odemeler FOREIGN KEY (oId)
REFERENCES Odemeler(oId)
);

-- Üyelik tablosu
CREATE TABLE Uyelik (
    uyelikId INT PRIMARY KEY IDENTITY(1,1),
    kisiId INT NOT NULL,
    oId INT,
    baslangicTarih DATE,
    bitisTarih DATE,
    CONSTRAINT fk_uyelik_kisiler FOREIGN KEY (kisiId) REFERENCES
Kisiler(kisiId),
    CONSTRAINT fk_uyelik_odemeler FOREIGN KEY (oId) REFERENCES
Odemeler(oId)
);

-- Antrenman tablosu
CREATE TABLE Antrenman (
    kisiId INT NOT NULL,
    eId INT NOT NULL,
    atmaTarihi DATE,
    PRIMARY KEY (kisiId, eId),
    CONSTRAINT fk_antrenman_kisiler FOREIGN KEY (kisiId) REFERENCES
Kisiler(kisiId),
    CONSTRAINT fk_antrenman_egitmenler FOREIGN KEY (eId) REFERENCES
Egitmenler(kisiId)
);
```

```
-- Sınav Merkezi tablosu
CREATE TABLE SinavMerkezi (
    sinavId INT PRIMARY KEY IDENTITY(1,1),
    kisiId INT NOT NULL,
    kemerId INT NOT NULL,
    oId INT,
    eId INT, -- Eğitmen kimliği sütunu
    sinavTarihi DATE,
    sonuc NVARCHAR(50),
    CONSTRAINT fk_sinavmerkezi_kisiler FOREIGN KEY (kisiId) REFERENCES
Kisiler(kisiId),
    CONSTRAINT fk_sinavmerkezi_kemer FOREIGN KEY (kemerId) REFERENCES
KemerSeviyesi(kemerId),
    CONSTRAINT fk_sinavmerkezi_odemeler FOREIGN KEY (oId) REFERENCES
Odemeler(oId),
    CONSTRAINT fk_sinavmerkezi_eId FOREIGN KEY (eId) REFERENCES
Egitmenler(kisiId)
);

-- Yarışmalar tablosu
CREATE TABLE Yarismalar (
    yId INT PRIMARY KEY IDENTITY(1,1),
    yarismaAdi NVARCHAR(100),
    yarismaTarihi DATE
);

-- Yarışma Katılım tablosu
CREATE TABLE YarismaKatilim (
    yId INT NOT NULL,
    kisiId INT NOT NULL,
    kazandi BIT NOT NULL,
    PRIMARY KEY (yId, kisiId),
    CONSTRAINT fk_yarismakatilim_yarismalar FOREIGN KEY (yId)
REFERENCES Yarismalar(yId),
    CONSTRAINT fk_yarismakatilim_kisiler FOREIGN KEY (kisiId) REFERENCES
Kisiler(kisiId)
);

-- Ödüller tablosu
CREATE TABLE Odulleri (
    odulId INT PRIMARY KEY IDENTITY(1,1),
    yId INT NOT NULL,
    odulAdi NVARCHAR(100),
    CONSTRAINT fk_odulleri_yarismalar FOREIGN KEY (yId) REFERENCES
Yarismalar(yId);
```

Triggerler :

```
-- Uyeler tablosuna veri eklendiğinde tetiklenecek bir tetikleyici (Trigger) oluşturuluyor.
CREATE TRIGGER CheckPersonRoleForUyeler
ON Uyeler
AFTER INSERT
AS
BEGIN
    -- Yeni eklenen kayıttaki kisiId değerini almak için bir değişken tanımlanıyor.
    DECLARE @kisiId INT;

    -- INSERTED tablosundan eklenen kaydın kisiId'si alınıyor.
    SELECT @kisiId = kisiId FROM INSERTED;

    -- Alınan kisiId'nin Egitmenler tablosunda zaten mevcut olup olmadığını kontrol ediyoruz.
    IF EXISTS (SELECT 1 FROM Egitmenler WHERE kisiId = @kisiId)
    BEGIN
        -- Eğer kisiId, Egitmenler tablosunda mevcutsa, hata mesajı gösterilir ve işlem geri alınır.
        PRINT 'Girilen kişi zaten eğitmen üye olamaz!!!';
        ROLLBACK;
        RETURN;
    END
END;
GO  -- Tetikleyici ve diğer SQL ifadeleri arasında ayrım yapmak için kullanılır.
```

Bu tetikleyicinin amacı:
Eğer kisiId değeri, Egitmenler tablosunda zaten bir eğitmene aitse, bu kişi Uyeler tablosuna üye olarak eklenemez. Bu durumda tetikleyici devreye girer, işlemi iptal eder ve şu mesajı gösterir:

"Girilen kişi zaten eğitmen üye olamaz!!!"

Bu, veri bütünlüğünü korumak ve aynı kişinin hem eğitmen hem de üye olarak kaydedilmesini engellemek için kullanılır.

```
-- Uyeler tablosuna veri eklendiğinde tetiklenecek bir tetikleyici (Trigger) oluşturuluyor.
CREATE TRIGGER CheckPersonRoleForUyeler
ON Uyeler
AFTER INSERT
AS
BEGIN
   -- Yeni eklenen kayıttaki kisiId değerini almak için bir değişken tanımlanıyor.
   DECLARE @kisiId INT;

   -- INSERTED tablosundan eklenen kaydın kisiId'si alınıyor.
   SELECT @kisiId = kisiId FROM INSERTED;

   -- Alınan kisiId'nin Egitmenler tablosunda zaten mevcut olup olmadığını kontrol ediyoruz.
   IF EXISTS (SELECT 1 FROM Egitmenler WHERE kisiId = @kisiId)
   BEGIN
      -- Eğer kisiId, Egitmenler tablosunda mevcutsa, hata mesajı gösterilir ve işlem geri alınır.
      PRINT 'Girilen kişi zaten eğitmen üye olamaz!!!';
      ROLLBACK;
      RETURN;
   END
END;
GO  -- Tetikleyici ve diğer SQL ifadeleri arasında ayrım yapmak için kullanılır.
```

Bu tetikleyicinin amacı:
Eğer kisiId değeri, Egitmenler tablosunda zaten bir eğitmene aitse, bu kişi Uyeler tablosuna üye olarak eklenemez. Bu durumda tetikleyici devreye girer, işlemi iptal eder ve şu mesajı gösterir:

"Girilen kişi zaten eğitmen üye olamaz!!!"

Bu, veri bütünlüğünü korumak ve aynı kişinin hem eğitmen hem de üye olarak kaydedilmesini engellemek için kullanılır.

```
-- Odemeler tablosuna yeni bir ödeme eklendiğinde tetiklenecek bir tetikleyici (Trigger) oluşturuluyor.
CREATE TRIGGER UpdateGenelBakiye
ON Odemeler
AFTER INSERT
AS
BEGIN
    -- Son ödeme öncesindeki genelBakiye değerini almak için bir değişken tanımlanıyor.
    DECLARE @lastGenelBakiye DECIMAL(15, 2);

    -- En son eklenen ödemeden bir önceki oId'ye ait genelBakiye değerini alıyoruz.
    SELECT @lastGenelBakiye = genelBakiye
    FROM Odemeler
    WHERE oId = (
        SELECT MAX(oId)
        FROM Odemeler
        WHERE oId < (SELECT MAX(oId) FROM Odemeler)  -- İkinci en büyük oId değerini alıyoruz.
    );

    -- Yeni eklenen ödeme miktarını (paraMiktari) genelBakiye'ye ekleyerek güncelliyoruz.
    UPDATE Odemeler
    SET genelBakiye = @lastGenelBakiye + (SELECT paraMiktari FROM INSERTED)
    WHERE oId = (SELECT oId FROM INSERTED);
END;
```

Bu tetikleyicinin amacı:
Bu tetikleyici, Odemeler tablosuna yeni bir ödeme eklendiğinde çalışır. Yeni eklenen ödemenin paraMiktari değeri, bir önceki genel bakiyeye (genelBakiye) eklenerek otomatik olarak güncellenir. Böylece her yeni ödeme işleminde genelBakiye değeri otomatik olarak güncellenmiş olur.

```
-- Uyelik tablosuna yeni bir üyelik eklendiğinde tetiklenecek bir tetikleyici (Trigger) oluşturuluyor.
CREATE TRIGGER trg_UpdateAktifMiOnUyelik
ON Uyelik
AFTER INSERT
AS
BEGIN
    -- Yeni eklenen üyelik bilgilerine göre Uyeler tablosundaki aktifMi sütununu 1 olarak güncelliyoruz.
    UPDATE Uyeler
    SET aktifMi = 1
    WHERE kisiId IN (
        SELECT kisiId
        FROM inserted
    );
END;
```

Bu tetikleyicinin amacı:
Bu tetikleyici, bir üyenin Uyelik tablosuna kaydedildiğinde, Uyeler tablosundaki aktifMi sütununu otomatik olarak 1 yapar. Böylece yeni üyelik oluşturan kişi aktif hale getirilmiş olur.

```
-- SinavMerkezi tablosuna veri eklenirken çalışacak bir tetikleyici (Trigger)
oluşturuluyor.
CREATE TRIGGER trg_PreventExamAndCreatePayment
ON SinavMerkezi
INSTEAD OF INSERT
AS
BEGIN
   DECLARE @kemerId INT;
   DECLARE @oId INT;
   DECLARE @paraMiktari FLOAT;

   -- Eklenen verilerden kemerId ve oId değerlerini alıyoruz.
   SELECT @kemerId = kemerId, @oId = oId FROM inserted;

   -- Eğer kemerId = 8 ise (siyah kemer seviyesi)
   IF @kemerId = 8
   BEGIN
      -- Kullanıcıya en yüksek seviyeye ulaştığını belirten bir hata mesajı gösterilir.
      RAISERROR ('Sınava giremezsiniz çünkü en yüksek kemer seviyesine ulaştınız.',
16, 1);

      -- Girilen oId için paraMiktari değerini alıyoruz.
      SELECT @paraMiktari = paraMiktari FROM Odemeler WHERE oId = @oId;

      -- Eğer paraMiktari pozitifse, bu değeri negatif yapıyoruz.
      IF @paraMiktari > 0
         SET @paraMiktari = -@paraMiktari;

      -- Negatif paraMiktari ile Odemeler tablosuna yeni bir ödeme kaydı ekliyoruz.
      INSERT INTO Odemeler (odemeTarihi, paraMiktari)
      VALUES (GETDATE(), @paraMiktari);
   END
   ELSE
   BEGIN
      -- Eğer kemerId 8 değilse, SinavMerkezi tablosuna veri normal şekilde eklenir.
      INSERT INTO SinavMerkezi (kisiId, kemerId, oId, eId, sinavTarihi, sonuc)
      SELECT kisiId, kemerId, oId, eId, sinavTarihi, sonuc FROM inserted;
   END
END;
```

Bu tetikleyicinin amacı:
Bu tetikleyici, siyah kemer seviyesine (kemerId = 8) ulaşmış bir üyenin sınava girmesini engeller. Eğer bu üye sınav ücreti ödemişse, bir hata mesajı gösterilir ve ödediği ücret negatif bir değer olarak Odemeler tablosuna geri ödeme kaydı eklenir. Siyah kemer seviyesine ulaşmamış üyeler için veri SinavMerkezi tablosuna normal --

SinavMerkezi tablosuna veri eklendikten sonra çalışacak bir tetikleyici (Trigger) oluşturuluyor.

```
CREATE TRIGGER trg_UpdateKemerId
ON SinavMerkezi
AFTER INSERT
AS
BEGIN
   DECLARE @kisiId INT;
   DECLARE @sonuc BIT;
   DECLARE @kemerId INT;

   -- Eklenen tablodan kisiId, sonuc ve kemerId değerlerini alıyoruz.
   SELECT @kisiId = kisiId, @sonuc = sonuc, @kemerId = kemerId FROM inserted;

   -- Eğer sınav sonucu başarılıysa (sonuc = 1)
   IF @sonuc = 1
   BEGIN
      -- Uyeler tablosundaki kemerId değerini 1 artırıyoruz.
      UPDATE Uyeler
      SET kemerId = @kemerId + 1
      WHERE kisiId = @kisiId;
   END
END;şekilde eklenir.
```

Bu tetikleyicinin amacı:
Bu tetikleyici, SinavMerkezi tablosuna yeni bir sınav kaydı eklendiğinde çalışır. Eğer sınav sonucu (sonuc) başarılıysa (1), ilgili üyenin Uyeler tablosundaki kemer seviyesi (kemerId) bir üst seviyeye yükseltilir.

```
-- YarismaKatilim tablosuna veri eklendikten sonra çalışacak bir tetikleyici (Trigger) oluşturuluyor.
CREATE TRIGGER trg_InsertOdul
ON YarismaKatilim
AFTER INSERT
AS
BEGIN
   -- kazandi = 1 durumunu kontrol ederek ödülü otomatik olarak Odulleri tablosuna ekliyoruz.
   INSERT INTO Odulleri (yId, odulAdi)
   SELECT
      i.yId,
      (SELECT yarismaAdi FROM Yarismalar WHERE Yarismalar.yId = i.yId)
   FROM
      INSERTED i
   WHERE
      i.kazandi = 1;
END;
```

Bu tetikleyicinin amacı:
Bu tetikleyici, YarismaKatilim tablosuna yeni bir yarışma katılım kaydı eklendiğinde çalışır. Eğer katılımcı yarışmayı kazanmışsa (kazandi = 1), ilgili yarışmanın adı Odulleri tablosuna ödül olarak otomatik şekilde eklenir.

Procedurler

```
-- Egitmenlerin bilgilerini getiren bir prosedür oluşturuluyor.
CREATE PROCEDURE EgitmenleriGetir
AS
BEGIN
   SELECT
      K.kisiId,
      K.adi AS Adi,
      K.soyadi AS Soyadi,
      K.telefon AS Telefon,
      K.eposta AS Eposta,
      K.adres AS Adres,
      E.yetenekleri AS Yetenekleri
   FROM
      Egitmenler E
   INNER JOIN
      Kisiler K ON E.kisiId = K.kisiId;
END;

EXEC EgitmenleriGetir;
```

```
-- Pasif üyelerin bilgilerini getiren bir prosedür oluşturuluyor.
CREATE PROCEDURE PasifUyeleriGetir
AS
BEGIN
   SELECT
      K.kisiId,
      K.adi AS Adi,
      K.soyadi AS Soyadi,
      K.telefon AS Telefon,
      K.eposta AS Eposta,
      K.adres AS Adres,
      KS.kemerRengi AS KemerSeviyesi
   FROM
      Uyeler U
   INNER JOIN
      Kisiler K ON U.kisiId = K.kisiId
   LEFT JOIN
      KemerSeviyesi KS ON U.kemerId = KS.kemerId
   WHERE
```

```
    U.aktifMi = 0 OR U.aktifMi IS NULL; -- Pasif veya durumu bilinmeyen üyeleri
getir
END;

EXEC PasifUyeleriGetir;


-- Aktif üyelerin bilgilerini getiren bir prosedür oluşturuluyor.
CREATE PROCEDURE AktifOgrencilerGetir
AS
BEGIN
  SELECT
    K.kisiId,
    K.adi AS Adi,
    K.soyadi AS Soyadi,
    K.telefon AS Telefon,
    K.eposta AS Eposta,
    K.adres AS Adres,
    U.aktifMi AS AktifDurumu,
    KS.kemerRengi AS KemerSeviyesi
  FROM
    Uyeler U
  INNER JOIN
    Kisiler K ON U.kisiId = K.kisiId
  LEFT JOIN
    KemerSeviyesi KS ON U.kemerId = KS.kemerId
  WHERE
    U.aktifMi = 1; -- Sadece aktif üyeleri getir
END;

EXEC AktifOgrencilerGetir;



-- Belirli bir kemer seviyesine sahip üyeleri getiren prosedür
CREATE PROCEDURE KemerIdIleUyeleriGetir
  @kemerId INT
AS
BEGIN
  SELECT
    K.kisiId,
    K.adi AS Adi,
    K.soyadi AS Soyadi,
    K.telefon AS Telefon,
    K.eposta AS Eposta,
    K.adres AS Adres,
```

```
        U.aktifMi AS AktifDurumu,
        KS.kemerRengi AS KemerRengi
    FROM
        Uyeler U
    INNER JOIN
        Kisiler K ON U.kisiId = K.kisiId
    INNER JOIN
        KemerSeviyesi KS ON U.kemerId = KS.kemerId
    WHERE
        U.kemerId = @kemerId; -- Belirtilen kemerId'ye sahip üyeleri getir
END;

EXEC KemerIdIleUyeleriGetir @kemerId=7;


-- Tüm yarışma katılımcılarını getiren prosedür
CREATE PROCEDURE GetAllYarismalarParticipants
AS
BEGIN
    SELECT
        YK.yId,
        Y.yarismaAdi AS YarismaAdi,
        Y.yarismaTarihi AS YarismaTarihi,
        K.kisiId,
        K.adi AS Adi,
        K.soyadi AS Soyadi,
        K.telefon AS Telefon,
        K.eposta AS Eposta,
        K.adres AS Adres,
        YK.kazandi AS Kazandi
    FROM
        YarismaKatilim YK
    INNER JOIN
        Yarismalar Y ON YK.yId = Y.yId
    INNER JOIN
        Kisiler K ON YK.kisiId = K.kisiId;
END;

EXEC GetAllYarismalarParticipants;



-- Belirli bir eğitmenin öğrencilerini getiren prosedür
CREATE PROCEDURE EgitmeninOgrencileriGetir
    @egitmenId INT
AS
```

```
BEGIN
    SELECT
        K.kisiId,
        K.adi AS Adi,
        K.soyadi AS Soyadi,
        K.telefon AS Telefon,
        K.eposta AS Eposta,
        K.adres AS Adres,
        KS.kemerRengi AS KemerSeviyesi
    FROM
        Antrenman A
    INNER JOIN
        Kisiler K ON A.kisiId = K.kisiId
    LEFT JOIN
        Uyeler U ON K.kisiId = U.kisiId
    LEFT JOIN
        KemerSeviyesi KS ON U.kemerId = KS.kemerId
    WHERE
        A.eId = @egitmenId; -- Belirli eğitmenin öğrencilerini getir
END;
EXEC EgitmeninOgrencileriGetir 38;


-- Belirli bir eğitmenin başarılı öğrencilerini getiren prosedür
CREATE PROCEDURE EgitmeninBasariliOgrencileriGetir
    @egitmenId INT
AS
BEGIN
    SELECT
        K.kisiId,
        K.adi AS Adi,
        K.soyadi AS Soyadi,
        K.telefon AS Telefon,
        K.eposta AS Eposta,
        K.adres AS Adres,
        KS.kemerRengi AS KemerSeviyesi
    FROM
        SinavMerkezi SM
    INNER JOIN
        Kisiler K ON SM.kisiId = K.kisiId
    LEFT JOIN
        KemerSeviyesi KS ON SM.kemerId = KS.kemerId
    WHERE
        SM.eId = @egitmenId
        AND SM.sonuc = 1; -- Sadece başarılı öğrencileri getir
END;
```

```
-- Yarışmayı kazananlar ve aldıkları ödülleri getiren prosedür
CREATE PROCEDURE GetWinnersAndAwards
AS
BEGIN
    SELECT
        K.kisiId,
        K.adi AS Adi,
        K.soyadi AS Soyadi,
        K.telefon AS Telefon,
        K.eposta AS Eposta,
        K.adres AS Adres,
        Y.yarismaAdi AS YarismaAdi,
        Y.yarismaTarihi AS YarismaTarihi,
        O.odulAdi AS OdulAdi
    FROM
        YarismaKatilim YK
    INNER JOIN
        Yarismalar Y ON YK.yId = Y.yId
    INNER JOIN
        Kisiler K ON YK.kisiId = K.kisiId
    LEFT JOIN
        Odulleri O ON Y.yId = O.yId
    WHERE
        YK.kazandi = 1; -- Yarışmayı kazananların bilgileri
END;

EXEC GetWinnersAndAwards;
```

Transaction :

```
CREATE PROCEDURE AddNewMemberToMembership
    @adi NVARCHAR(50),
    @soyadi NVARCHAR(50),
    @telefon NVARCHAR(20),
    @eposta NVARCHAR(50),
    @adres NVARCHAR(255),
    @paraMiktari FLOAT -- Ödeme miktarı
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;  -- İşlem başlat

        -- 1. Yeni üyeyi Kisiler tablosuna ekleyin
        DECLARE @kisiId INT;
        INSERT INTO Kisiler (adi, soyadi, telefon, eposta, adres)
        VALUES (@adi, @soyadi, @telefon, @eposta, @adres);

        SET @kisiId = SCOPE_IDENTITY(); -- Yeni eklenen üyenin ID'sini alın

        -- Üye ekleme başarılı mı kontrol et
        IF @kisiId IS NULL
        BEGIN
            THROW 50000, 'Yeni üye eklenemedi.', 1;
        END

        -- 2. Ödeme ekleyin
        DECLARE @oId INT;
        INSERT INTO Odemeler (odemeTarihi, paraMiktari)
        VALUES (GETDATE(), @paraMiktari); -- Ödemeyi kaydedin

        SET @oId = SCOPE_IDENTITY(); -- Ödeme ID'sini alın

        -- Ödeme ekleme başarılı mı kontrol et
        IF @oId IS NULL
        BEGIN
            THROW 50000, 'Ödeme eklenemedi.', 1;
        END

        -- 3. Üyelik ekleyin
        DECLARE @baslangicTarih DATE = GETDATE();
        DECLARE @bitisTarih DATE = DATEADD(YEAR, 1, GETDATE()); -- Üyelik
süresi bir yıl
        INSERT INTO Uyelik (kisiId, oId, baslangicTarih, bitisTarih)
```

```
        VALUES (@kisiId, @oId, @baslangicTarih, @bitisTarih);

        -- Üyelik ekleme başarılı mı kontrol et
        IF NOT EXISTS (SELECT 1 FROM Uyelik WHERE kisiId = @kisiId AND oId
= @oId)
        BEGIN
            THROW 50000, 'Üyelik eklenemedi.', 1;
        END

        -- Tüm işlemler başarılıysa işlemi onaylayın
        COMMIT TRANSACTION;

    END TRY
    BEGIN CATCH
        -- Hata durumunda işlemi geri alın
        ROLLBACK TRANSACTION;

        -- Hata mesajını göster
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();

        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END;

EXEC AddNewMemberToMembership
    @adi = 'beshar',
    @soyadi = 'elhatib',
    @telefon = '05011234567',
    @eposta = 'biso.veli@example.com',
    @adres = 'suriye',
    @paraMiktari = 16000.00;

SELECT * FROM Odemeler;
SELECT * FROM Kisiler;
```

Burada, üyelik işlemleri ve ödeme işlemleri arasında bir işlem (transaction) kullandık. Bu sayede, eğer ödeme yapılırsa, genel bakiye güncellenir. Ayrıca, üyelik aktif hale

getirilirse, üye de aktif olur. Bu işlemler birbirine bağlı olduğundan, bir işlem başarısız olursa, tüm işlem geri alınır (ROLLBACK).

Veriler Girişi :

```
INSERT INTO Kisiler (adi, soyadi, telefon, eposta, adres)
VALUES
('Ahmet', 'Yılmaz', '5551000000', 'ahmet1@example.com', 'İstanbul, Türkiye'),
('Mehmet', 'Kaya', '5551000001', 'mehmet1@example.com', 'Ankara, Türkiye'),
('Ali', 'Şahin', '5551000002', 'ali1@example.com', 'Bursa, Türkiye'),
('Hasan', 'Öztürk', '5551000003', 'hasan1@example.com', 'Adana, Türkiye'),
('Kemal', 'Çelik', '5551000004', 'kemal1@example.com', 'Eskişehir, Türkiye'),
('Hüseyin', 'Demir', '5551000005', 'huseyin1@example.com', 'Konya, Türkiye'),
('Murat', 'Koç', '5551000006', 'murat1@example.com', 'Gaziantep, Türkiye'),
('İsmail', 'Yıldız', '5551000007', 'ismail1@example.com', 'Samsun, Türkiye'),
('Mehmet', 'Arslan', '5551000008', 'mehmet2@example.com', 'Bolu, Türkiye'),
('Cem', 'Güney', '5551000009', 'cem1@example.com', 'Kayseri, Türkiye'),
('Selim', 'Uysal', '5551000010', 'selim1@example.com', 'Mersin, Türkiye'),
('Fatih', 'Aydın', '5551000011', 'fatih1@example.com', 'Malatya, Türkiye'),
('Ömer', 'Kara', '5551000012', 'omer1@example.com', 'Denizli, Türkiye'),
('Veli', 'Günay', '5551000013', 'veli1@example.com', 'Sakarya, Türkiye'),
('Sefa', 'Özdemir', '5551000014', 'sefa1@example.com', 'Ordu, Türkiye'),
('Tuncay', 'Gündoğdu', '5551000015', 'tuncay1@example.com', 'Trabzon, Türkiye'),
('Serkan', 'Kılıç', '5551000016', 'serkan1@example.com', 'Burdur, Türkiye'),
('Hikmet', 'Aslan', '5551000017', 'hikmet1@example.com', 'Çorum, Türkiye'),
('Rıza', 'Demirtaş', '5551000018', 'riza1@example.com', 'Zonguldak, Türkiye'),
('Yusuf', 'Karaaslan', '5551000019', 'yusuf1@example.com', 'Tekirdağ, Türkiye'),
('Ahmet', 'Yılmaz', '5551000020', 'ahmet2@example.com', 'İzmir, Türkiye'),
('Mehmet', 'Kaya', '5551000021', 'mehmet2@example.com', 'Antalya, Türkiye'),
('Ali', 'Şahin', '5551000022', 'ali2@example.com', 'Eskişehir, Türkiye'),
('Hasan', 'Öztürk', '5551000023', 'hasan2@example.com', 'Trabzon, Türkiye'),
('Kemal', 'Çelik', '5551000024', 'kemal2@example.com', 'Adana, Türkiye'),
('Hüseyin', 'Demir', '5551000025', 'huseyin2@example.com', 'Bursa, Türkiye'),
('Murat', 'Koç', '5551000026', 'murat2@example.com', 'Samsun, Türkiye'),
('İsmail', 'Yıldız', '5551000027', 'ismail2@example.com', 'Mersin, Türkiye'),
('Mehmet', 'Arslan', '5551000028', 'mehmet3@example.com', 'Kocaeli, Türkiye'),
('Cem', 'Güney', '5551000029', 'cem2@example.com', 'Şanlıurfa, Türkiye'),
('Selim', 'Uysal', '5551000030', 'selim2@example.com', 'Kayseri, Türkiye'),
('Fatih', 'Aydın', '5551000031', 'fatih2@example.com', 'Ankara, Türkiye'),
('Ömer', 'Kara', '5551000032', 'omer2@example.com', 'Bolu, Türkiye'),
('Veli', 'Günay', '5551000033', 'veli2@example.com', 'Gaziantep, Türkiye'),
('Sefa', 'Özdemir', '5551000034', 'sefa2@example.com', 'İzmir, Türkiye'),
('Tuncay', 'Gündoğdu', '5551000035', 'tuncay2@example.com', 'Aydın, Türkiye'),
```

('Serkan', 'Kılıç', '5551000036', 'serkan2@example.com', 'Manisa, Türkiye'),
('Hikmet', 'Aslan', '5551000037', 'hikmet2@example.com', 'Denizli, Türkiye'),
('Rıza', 'Demirtaş', '5551000038', 'riza2@example.com', 'Konya, Türkiye'),
('Yusuf', 'Karaaslan', '5551000039', 'yusuf2@example.com', 'Bursa, Türkiye');

INSERT INTO KemerSeviyesi (kemerRengi)
VALUES
('Beyaz Kemer'),
('Sarı Kemer'),
('Turuncu Kemer'),
('Yeşil Kemer'),
('Mavi Kemer'),
('Mor Kemer'),
('Kahverengi Kemer'),
('Siyah Kemer');

INSERT INTO Egitmenler (kisiId, yetenekleri)
VALUES
(1, 'Savaş teknikleri , geleneksel dövüş sanatları'),
(2, 'Temel hareketler eğitimi, savunma teknikleri, hareket analizi'),
(3, 'Savunma taktiği, dövüş analizi'),
(22, 'Fiziksel eğitim, hızlı tepki teknikleri, ileri dövüş teknikleri'),
(32, 'Hücum taktiği, karmaşık hareket eğitimi'),
(33, 'Denge çalışması, hareket koordinasyonu, ileri kata teknikleri'),
(38, 'Hızlı hareket eğitimi, kas gücü geliştirme');

INSERT INTO Uyeler (kisiId, aktifMi, kemerId)
VALUES
(4, 0, 4),
(5, 0, 5),
(6, 0, 6),
(7, 0, 7),
(8, 0, 1),
(9, 0, 2),
(10, 0, 3),
(11, 0, 4),
(12, 0, 5),
(13, 0, 6),
(14, 0, 7),
(15, 0, 1),
(16, 0, 2),

(17, 0, 3),
(18, 0, 4),
(19, 0, 5),
(20, 0, 6),
(21, 0, 7),
(23, 0, 2),
(24, 0, 3),
(25, 0, 4),
(26, 0, 5),
(27, 0, 6),
(28, 0, 7),
(29, 0, 1),
(30, 0, 2),
(31, 0, 3),
(34, 0, 4),
(35, 0, 5),
(36, 0, 1),
(37, 0, 1),
(39, 0, 2),
(40, 0, 3);

select * from Uyeler;

DECLARE @lastGenelBakiye DECIMAL(15, 2);

INSERT INTO Odemeler (odemeTarihi, paraMiktari)
VALUES
('2026-01-03', 2000);

select * from Odemeler;

INSERT INTO EgitmenKontrati (egitmenId, oId, kontratBaslangicTarihi,
kontratBitisTarihi)
VALUES
(38, 52, '2026-01-03', '2028-01-03'),
(33, 47, '2026-02-01', '2029-02-01'),
(3, 5, '2024-03-01', '2028-03-01'),
(22, 27, '2024-01-01', '2028-01-01');

```
select * from EgitmenKontrati;


select * from Egitmenler;

select* from Uyeler;

INSERT INTO Uyelik (kisiId, oId, baslangicTarih, bitisTarih)
VALUES
(5, 6, '2024-01-01', '2026-01-01'),
(6, 7, '2024-02-01', '2026-02-01'),
(7, 8, '2024-03-01', '2025-03-01'),
(8, 9, '2024-04-01', '2025-04-01'),
(9, 10, '2024-05-01', '2025-05-01'),
(10, 11, '2024-06-01', '2025-06-01'),
(11, 12, '2024-07-01', '2025-07-01'),
(12, 13, '2024-08-01', '2027-08-01'),
(19, 14, '2024-09-01', '2028-09-01'),
(20, 15, '2024-10-01', '2025-10-01'),
(21, 16, '2024-11-01', '2026-11-01'),
(4, 17, '2024-12-01', '2026-12-01'),
(13, 18, '2024-01-01', '2026-01-01'),
(14, 19, '2024-02-01', '2025-02-01'),
(15, 24, '2024-03-01', '2025-03-01'),
(16, 21, '2024-04-01', '2025-04-01'),
(17, 25, '2024-05-01', '2025-05-01'),
(18, 26, '2024-06-01', '2025-06-01');

select * from Uyelik;


INSERT INTO Uyelik (kisiId, oId, baslangicTarih, bitisTarih)
VALUES
(23, 37, '2024-01-01', '2026-01-01'),
(24, 38, '2024-02-01', '2026-02-01'),
(25, 39, '2024-03-01', '2025-03-01'),
(26, 40, '2024-04-01', '2025-04-01'),
(27, 41, '2024-05-01', '2025-05-01'),
(28, 42, '2024-06-01', '2025-06-01'),
(29, 43, '2024-07-01', '2025-07-01'),
(30, 44, '2024-08-01', '2027-08-01'),
(31, 45, '2024-08-01', '2027-08-01');
```

```
INSERT INTO Uyelik (kisiId, oId, baslangicTarih, bitisTarih)
VALUES
(34, 48, '2026-01-01', '2027-01-01'),
(35, 49, '2026-02-01', '2027-02-01'),
(36, 50, '2026-03-01', '2027-03-01'),
(37, 51, '2026-04-01', '2027-04-01');

select * from Uyeler;

INSERT INTO Uyelik (kisiId, oId, baslangicTarih, bitisTarih)
VALUES
(39, 53, '2026-01-01', '2027-01-01'),
(40, 54, '2026-02-01', '2027-02-01'),
(36, 50, '2026-03-01', '2027-03-01'),
(37, 51, '2026-04-01', '2027-04-01');


 INSERT INTO Antrenman (kisiId, eId)
VALUES
(4, 1),
(5, 2),
(6, 3),
(7, 22),
(8, 32),
(9, 33),
(10, 38),
(11, 1),
(12, 2),
(13, 3),
(14, 22),
(15, 32),
(16, 33),
(17, 38),
(18, 1),
(19, 2),
(20, 3),
(21, 22),
(23, 32),
(24, 33),
(25, 38),
(26, 1),
(27, 2),
(28, 3),
```

(29, 22),
(30, 32),
(31, 33),
(34, 38),
(35, 1),
(36, 2),
(37, 3),
(39, 22),
(40, 32),
(4, 33),
(5, 38),
(6, 1),
(7, 2),
(8, 3),
(9, 22),
(10, 32);

INSERT INTO Antrenman (kisiId, eId)
VALUES
(11, 33),
(12, 38),
(13, 1),
(14, 2),
(15, 3),
(16, 22),
(17, 32),
(18, 33),
(19, 38),
(20, 1),
(21, 2),
(23, 3),
(24, 22),
(25, 32),
(26, 33),
(27, 38),
(28, 1),
(29, 2),
(30, 3),
(31, 22),
(34, 32),
(35, 33),
(36, 38),
(37, 1),
(39, 2),
(40, 3),

(4, 22),
(5, 32),
(6, 33),
(7, 38),
(8, 1),
(9, 2),
(10, 3),
(11, 22),
(12, 32),
(13, 33),
(14, 38),
(15, 1),
(16, 2),
(17, 3),
(18, 22);

INSERT INTO Antrenman (kisiId, eId)
VALUES
(4, 38),
(5, 1),
(6, 2),
(7, 3),
(8, 22),
(9, 32),
(10, 33),
(11, 38),
(12, 1),
(13, 2),
(14, 3),
(15, 22),
(16, 32),
(17, 33),
(18, 38),
(19, 1),
(20, 2),
(21, 3),
(23, 22),
(24, 32),
(25, 33),
(26, 38),
(27, 1),
(28, 2),
(29, 3),
(30, 22),
(31, 32),

(34, 33),
(35, 38),
(36, 1),
(37, 2),
(39, 3),
(40, 22),
(4, 32),
(5, 33),
(6, 38),
(7, 1),
(8, 2),
(9, 3);

select * from Antrenman;

INSERT INTO SinavMerkezi (kisiId, kemerId, oId, eId, sinavTarihi, sonuc)
VALUES (39,3 ,1095 , 38, '2026-03-05', 0);

INSERT INTO Odemeler (odemeTarihi, paraMiktari)
VALUES
('2026-02-05', 12000);

INSERT INTO Yarismalar (yarismaAdi, yarismaTarihi)
VALUES
('Gaziantep Olimpiyatları', '2025-01-10'),
('Kilis Olimpiyatları', '2025-01-15'),
('Şanlıurfa Olimpiyatları', '2025-01-20'),
('Adana Olimpiyatları', '2025-01-25'),
('Hatay Olimpiyatları', '2025-02-01'),
('Mersin Olimpiyatları', '2025-02-10'),
('Antalya Olimpiyatları', '2025-02-15'),
('İzmir Olimpiyatları', '2025-02-20'),
('Bursa Olimpiyatları', '2025-03-01'),
('Konya Olimpiyatları', '2025-03-10'),
('Ankara Olimpiyatları', '2025-03-15'),
('İstanbul Olimpiyatları', '2025-03-20'),
('Trabzon Olimpiyatları', '2025-03-25'),

('Diyarbakır Olimpiyatları', '2025-04-01'),
('Van Olimpiyatları', '2025-04-10');

```
select * from Yarismalar;
select * from  Uyeler;
select * from  KemerSeviyesi;
select * from  Odemeler;
select * from  SinavMerkezi;
select * from Egitmenler;
select * from Odulleri;
select * from YarismaKatilim;
```

```
INSERT INTO YarismaKatilim (yId, kisiId, kazandi)
VALUES
(2, 8, 0),
(3, 9, 0),
(4, 10,1),
(5, 11, 0),
(6, 12, 0),
(7, 13, 0),
(8, 14, 0),
(9, 15, 0),
(10, 16, 1),
(11, 17, 0),
(12, 18, 0),
(13, 19, 0),
(14, 20, 0),
(15, 21, 1);
```

Triggerler denemek için komutlar :

```
use karateKlubu;

INSERT INTO Kisiler (adi, soyadi, telefon, eposta, adres)
VALUES
('Kerem', 'Demir', '5552000000', 'kerem@example.com', 'İzmir, Türkiye'),
('Zezo', 'Çelik', '5552000001', 'zeynep@example.com', 'Antalya, Türkiye'),
('Emin', 'Yıldız', '5552000002', 'elif@example.com', 'Konya, Türkiye'),
('Murat', 'Koç', '5552000003', 'murat@example.com', 'Gaziantep, Türkiye');
```

```
select * from Kisiler;

INSERT INTO Egitmenler (kisiId, yetenekleri)
VALUES
(43, 'Esneklik geliştirme, nefes kontrolü, meditasyon teknikleri'),
(44, 'Kuvvet antrenmanı, dayanıklılık geliştirme, ileri vuruş teknikleri');


INSERT INTO Uyeler (kisiId, aktifMi, kemerId)
VALUES
(45, 0, 1),
(46, 0, 2),
(43, 0, 2); --TRIGGER CheckPersonRoleForEgitmenler



INSERT INTO Egitmenler (kisiId, yetenekleri)
VALUES
(46, 'Konsantrasyon geliştirme, stratejik dövüş planlama, grup dövüş eğitimi'); --
TRIGGER CheckPersonRoleForEgitmenler



INSERT INTO Odemeler (odemeTarihi, paraMiktari)
VALUES
('2026-01-09', -90000); --TRIGGER UpdateGenelBakiye

INSERT INTO Odemeler (odemeTarihi, paraMiktari)
VALUES
('2026-01-09', -90000); --TRIGGER UpdateGenelBakiye

select * from Odemeler;
select * from Egitmenler;

INSERT INTO EgitmenKontrati (egitmenId, oId, kontratBaslangicTarihi,
kontratBitisTarihi)
VALUES
(43, 52, '2026-01-03', '2028-01-03'), -- oId select sonradan eklcez
(44, 27, '2024-01-01', '2028-01-01'); -- oId select sonradan eklcez


INSERT INTO Odemeler (odemeTarihi, paraMiktari)
VALUES
('2026-01-010', 9000); --TRIGGER UpdateGenelBakiye

INSERT INTO Odemeler (odemeTarihi, paraMiktari)
```

```
VALUES
('2026-01-010', 9000); --TRIGGER UpdateGenelBakiye

select * from Odemeler;

INSERT INTO Uyelik (kisiId, oId, baslangicTarih, bitisTarih)
VALUES
(45, 6, '2024-01-01', '2026-01-01'),  -- TRIGGER trg_UpdateAktifMiOnUyelik
(46, 7, '2024-02-01', '2026-02-01'); --  TRIGGER trg_UpdateAktifMiOnUyelik


INSERT INTO Odemeler (odemeTarihi, paraMiktari)
VALUES
('2026-01-11', 9000); --TRIGGER UpdateGenelBakiye

INSERT INTO Odemeler (odemeTarihi, paraMiktari)
VALUES
('2026-01-11', 9000); --TRIGGER UpdateGenelBakiye

select * from Uyeler;


INSERT INTO SinavMerkezi (kisiId, kemerId, oId, eId, sinavTarihi, sonuc)
VALUES (7,8 ,1095 , 38, '2026-03-05', 0); --TRIGGER
trg_PreventExamAndCreatePayment  --TRIGGER UpdateGenelBakiye

INSERT INTO SinavMerkezi (kisiId, kemerId, oId, eId, sinavTarihi, sonuc)
VALUES (9,3 ,1095 , 38, '2026-03-05', 1); -- TRIGGER trg_UpdateKemerId

INSERT INTO Yarismalar (yarismaAdi, yarismaTarihi)
VALUES
('HALEP  Olimpiyatları', '2025-01-10');

select * from Yarismalar;

INSERT INTO YarismaKatilim (yId, kisiId, kazandi)
VALUES
(16, 23, 1), --TRIGGER trg_InsertOdul
```