FIRAT ÜNİVERSİTESİ

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

BMÜ-329 VERİ TABANI SİSTEMLERİ
DÖNEM PROJESİ RAPORU

[Akıllı Tarım ve Sulama Yönetim Sistemi Veri Tabanı Projesi]

Grup No	7. Grup
Takım Üyesi 1	235260083 - Taha Buğra Çiçek
Takım Üyesi 2	235260035 - Ali Kağan Akdemir
Takım Üyesi 3	235260029 - Emircan Özkaya
Ders Sorumlusu	Prof. Dr. Burhan Ergen
Teslim Tarihi	[27/12/2025

İÇİNDEKİLER

1. GİRİŞ VE PROJE TANIMI

1.1 Projenin Amacı ve Kapsamı

Bu projenin temel amacı, tarımsal faaliyetleri dijital bir sisteme taşıyarak verimliliği artırmak ve kaynak yönetimini modernize etmektir. Proje kapsamında; tarlalardan alınan nem, sıcaklık ve toprak analizi gibi sensör verilerinin sistematik bir şekilde depolanması hedeflenmektedir.

Bu sayede sulama ve gübreleme gibi kritik işlemlerin en doğru zamanda yapılması sağlanarak su ve gübre israfının önüne geçilmesi planlanmaktadır. Hazırlanan ilişkisel veritabanı yapısı, farklı bölgelerdeki arazilerin durumunu tek bir merkezden izlemeye olanak tanır.

Ayrıca sistemde tutulan geçmiş hasat verileri, çiftçilerin bir sonraki ekim dönemi için daha doğru kararlar almasına yardımcı olur.

1.2 Hedef Kullanıcılar

Bu sistemin öncelikli kullanıcıları, teknolojiyi tarıma entegre ederek verimliliklerini artırmak isteyen modern çiftçiler ve tarımsal işletme sahipleridir. Bunun yanı sıra, sahadan gelen sensör verilerini analiz ederek teknik kararlar alan ziraat mühendisleridir. Bunlara ilaveten bu sistemden ilgili personeller ve işlerini uzaktan yöneten patronların da istifade etmesi hedeflenmiştir.

2. PROJE GEREKSİNİMLERİ

2.1 Fonksiyonel Gereksinimler

ID	Gereksinim Adı	Açıklama
FR-01	Otomatik Kritik Seviye Uyarı Sistemi	Sistem sensörlerden gelen verileri inceleyerek kritik değerler için reaksiyon vermesi (örnek: nem oranı).

2.2 Fonksiyonel Olmayan Gereksinimler

ID	Gereksinim Adı	Açıklama
NFR-01	Hata Yönetimi	Sistem, veritabanı seviyesinde oluşabilecek hataları yakalayabilmeli ve kullanıcıya/sisteme anlaşılır bir hata mesajı döndürmelidir.
NFR-02	Veri Bütünlüğü ve Atomiklik	Kritik veri tabanı işlemleri "ya hep ya hiç" prensibine göre çalışmalıdır. İşlemin herhangi bir adımında (örneğin olmayan bir çiftçi ID'si girilmesi durumunda) hata oluşursa, yapılan tüm değişiklikler geri alınmalı (Rollback) ve veri tabanı tutarlı eski haline döndürülmelidir.
NFR-03	Gerçek Zamanlı Tepki	Kritik durumların tespiti (örneğin toprağın kurumasının algılanması), verinin veritabanına yazıldığı anda ("After Insert") gecikmesiz olarak yapılmalıdır.

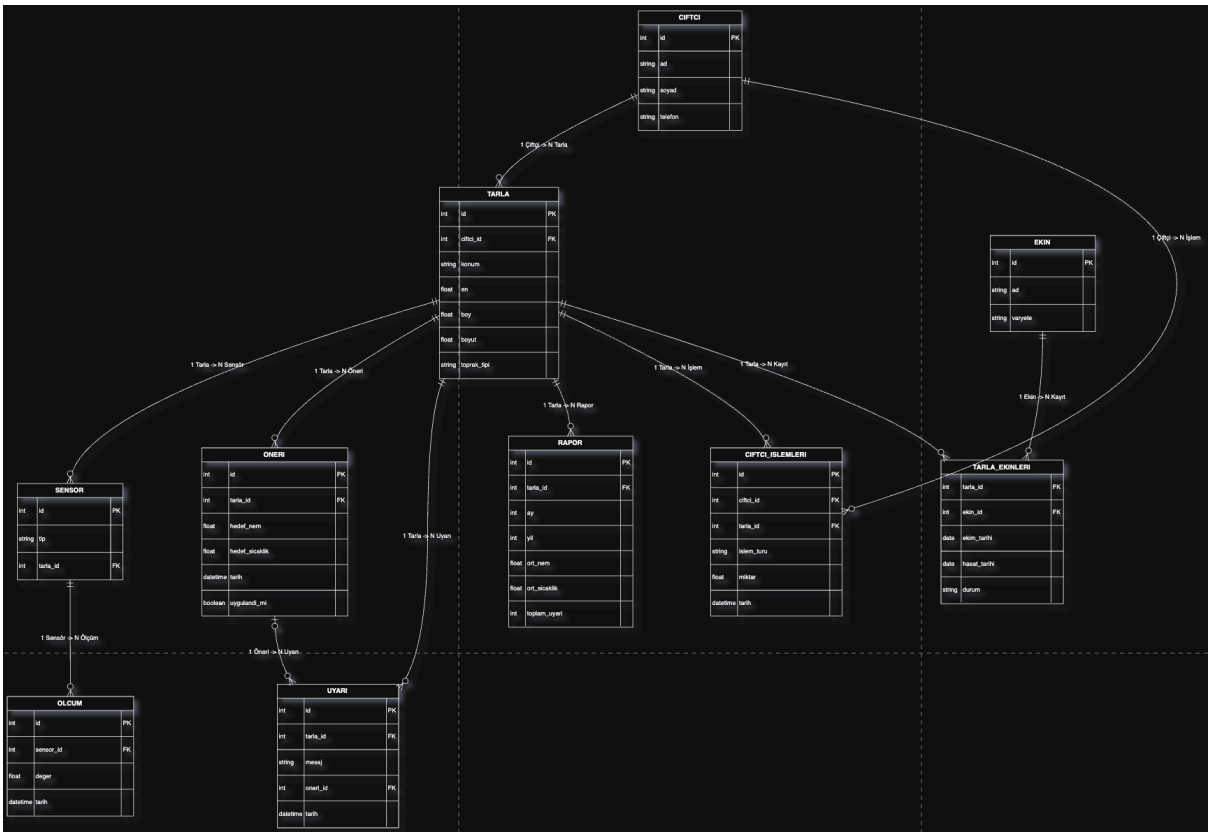
2.2 İş Kuralları, Kısıtlamalar

İş mantığı olarak sensörlerden gelen verilerin kritik değerler de olması halinde heme uyarı vermesi ve reaksiyon göstermesi. Birbiriyle ilişkili varlıkları ilgilendiren ortak verilerin her varlık nezin de ulaşılabilir ve kaydedilebilir olması aksi verilerin silinmesi veya uyarı mesajı verilmesi.

Kısıtlama olarak veri tabanında olmayan bir sensörden veri alınamaması, benzersizlik durumları ve tarih bilgisinin doğru veya eksiksiz girilmesi.

3. VERİ TABANI TASARIMI - VARLIK-İLİŞKİ (E-R) MODELİ

3.1 E-R Diyagramı



3.2 E-R Diyagramı Açıklaması

Akıllı Tarım ve Sulama Yönetim Sistemi için tasarlanan veri tabanı; çiftçiler, tarlalar, ekinler, sensörler ve sistem tarafından üretilen ölçüm, öneri ve raporları kapsayacak şekilde yapılandırılmıştır. Aşağıda sistemde yer alan tablolar ve bu tabloların temel alanları açıklanmaktadır.

CIFTCI (Çiftçi)
Çiftçilere ait temel bilgileri tutan tablodur.
Alanlar:
id (Primary Key)
ad
soyad
telefon_numarasi

EKIN (Ekin)
Tarım alanlarında yetiştirilen ürün bilgilerini tutar.
Alanlar:
id (Primary Key)
ad
varyete

TARLA (Tarla)
Çiftçilere ait tarla bilgilerini içeren tablodur.
Alanlar:
id (Primary Key)
ciftci_id (Foreign Key – CIFTCI.id)
konum
tarla_en
tarla_boy
tarla_boyut
toprak_tipi

TARLA_EKINLERI (Tarla Ekinleri)
Tarlalar ile ekinler arasındaki çoktan çoğa ilişkiyi temsil eden ilişki tablosudur.
Alanlar:
tarla_id (Primary Key, Foreign Key – TARLA.id)
ekin_id (Primary Key, Foreign Key – EKIN.id)

ekim_tarihi
beklenen_hasat_tarihi
durum

SENSOR (Sensör)
Tarlalarda bulunan sensörlere ait bilgileri tutar.
Alanlar:
id (Primary Key)
sensor_tipi
tarla_id (Foreign Key – TARLA.id)

OLCUM (Ölçüm)
Sensörlerden elde edilen ölçüm verilerini kaydeder.
Alanlar:
id (Primary Key)
sensor_id (Foreign Key – SENSOR.id)
deger
tarih

ONERI (Öneri)
Sistem tarafından tarlalar için oluşturulan sulama ve bakım önerilerini içerir.
Alanlar:
id (Primary Key)
tarla_id (Foreign Key – TARLA.id)
hedef_nem
hedef_sicaklik
hedef_ph
hedef_su_miktari
hedef_gubre_miktari
tarih
uygulandi_mi

UYARI (Uyarı)
Kritik durumlarda sistem tarafından oluşturulan uyarıları tutar.

Alanlar:
id (Primary Key)
tarla_id (Foreign Key – TARLA.id)
mesaj
oneri_id (Foreign Key – ONERI.id)
tarih

RAPOR (Rapor)
Tarlalara ait aylık ve yıllık özet rapor bilgilerini içerir.
Alanlar:
id (Primary Key)
tarla_id (Foreign Key – TARLA.id)
ay
yil
ortalama_nem
ortalama_sicaklik
ortalama_ph
toplam_uyari
basari_orani

CIFTCI_ISLEMLERI (Çiftçi İşlemleri)
Çiftçilerin tarlalar üzerinde gerçekleştirdiği işlemleri kayıt altına alır.
Alanlar:
id (Primary Key)
ciftci_id (Foreign Key – CIFTCI.id)
tarla_id (Foreign Key – TARLA.id)
islem_turu
miktar
birim
tarih
aciklama

4. İLİŞKİSEL ŞEMALAR

4.1 E-R'dan İlişkisel Şemalarına Dönüşüm

CIFTCI (<u>id</u>, ad, soyad, telefon_numarasi)

EKIN (<u>id</u>, ad, varyete)

TARLA (<u>id</u>, *ciftci_id*, konum, tarla_en, tarla_boy, tarla_boyut, toprak_tipi)

FK: `ciftci_id` → CIFTCI(id)

TARLA_EKINLERI (Çoka-Çok İlişki Tablosu) (<u>*tarla_id*</u>, <u>*ekin_id*</u>, ekim_tarihi, beklenen_hasat_tarihi, durum)

PK: (`tarla_id` + `ekin_id`) (Birleşik Anahtar)

FK: `tarla_id` → TARLA(id)

FK: `ekin_id` → EKIN(id)

SENSOR (<u>id</u>, sensor_tipi, *tarla_id*)

FK: `tarla_id` → TARLA(id)

OLCUM (<u>id</u>, *sensor_id*, deger, tarih)

FK: `sensor_id` → SENSOR(id)

ONERI (<u>id</u>, *tarla_id*, hedef_nem, hedef_sicaklik, hedef_ph, hedef_su_miktari, hedef_gubre_miktari, tarih, uygulandi_mi)

FK: `tarla_id` → TARLA(id)

UYARI (<u>id</u>, *tarla_id*, mesaj, *oneri_id*, tarih)

FK: `tarla_id` → TARLA(id)

FK: `oneri_id` → ONERI(id)

RAPOR (<u>id</u>, *tarla_id*, ay, yil, ortalama_nem, ortalama_sicaklik, ortalama_ph, toplam_uyari, basari_orani)

FK: `tarla_id` → TARLA(id)

CIFTCI_ISLEMLERI (<u>id</u>, *ciftci_id*, *tarla_id*, islem_turu, miktar, birim, tarih, aciklama)

- **FK:** `ciftci_id` → CIFTCI(id)
- **FK:** `tarla_id` → TARLA(id)

5. NORMALİZASYON

5.1 Normalizasyon Süreci

[Mümkünse BCNF değilse 3NF'e normalizasyon sürecini adım adım gösteriniz.]

tarim.CIFTCI tablosunda ad ve soyad ayrı sütunlarda tuttuk. Eğer "Ad Soyad" şeklinde tek sütun olsaydı, ileride sadece soyada göre sıralama yapmak zorlaşırdı.

Bir tarlada birden fazla ekin olabilir. Bunu TARLA tablosu içine ekin1, ekin2, ekin3 diye sütunlar ekleyerek çözmek yerine, tarim.TARLA_EKINLERI adında bir ara tablo oluşturarak çözüme kavuşturduk bu durumu da.

Eğer ölçüm değerlerini (sıcaklık, nem vb.) doğrudan SENSOR tablosuna yazsaydık, sensörün tipi değişmediği halde sürekli aynı sensör bilgisi tekrar edecekti.

Bunun yerine OLCUM tablosunda sadece sensor_id tutarak; "Ölçüm değeri → Sensör → Tarla" şeklindeki geçişli yapıyı tabloları ayırarak normalize ettik.

5.2 Son Normalize Edilmiş Şema

CIFTCI (id (PK), ad, soyad, telefon_numarasi)
EKIN (id (PK), ad, varyete)

TARLA (id (PK), ciftci_id (FK))

TARLA_EKINLERI (tarla_id (FK), ekin_id (FK))

SENSOR (id (PK), sensor_tipi, tarla_id (FK))

OLCUM (id (PK), sensor_id (FK))

ONERI (id (PK), tarla_id (FK))

UYARI (id (PK), tarla_id (FK), oneri_id (FK))

RAPOR (id (PK), tarla_id (FK))

CIFTCI_ISLEMLERI (id (PK), ciftci_id (FK), tarla_id (FK))

6. SQL SERVER VERİ TABANI ŞEMASI

6.1 Tablo Oluşturma Komutları

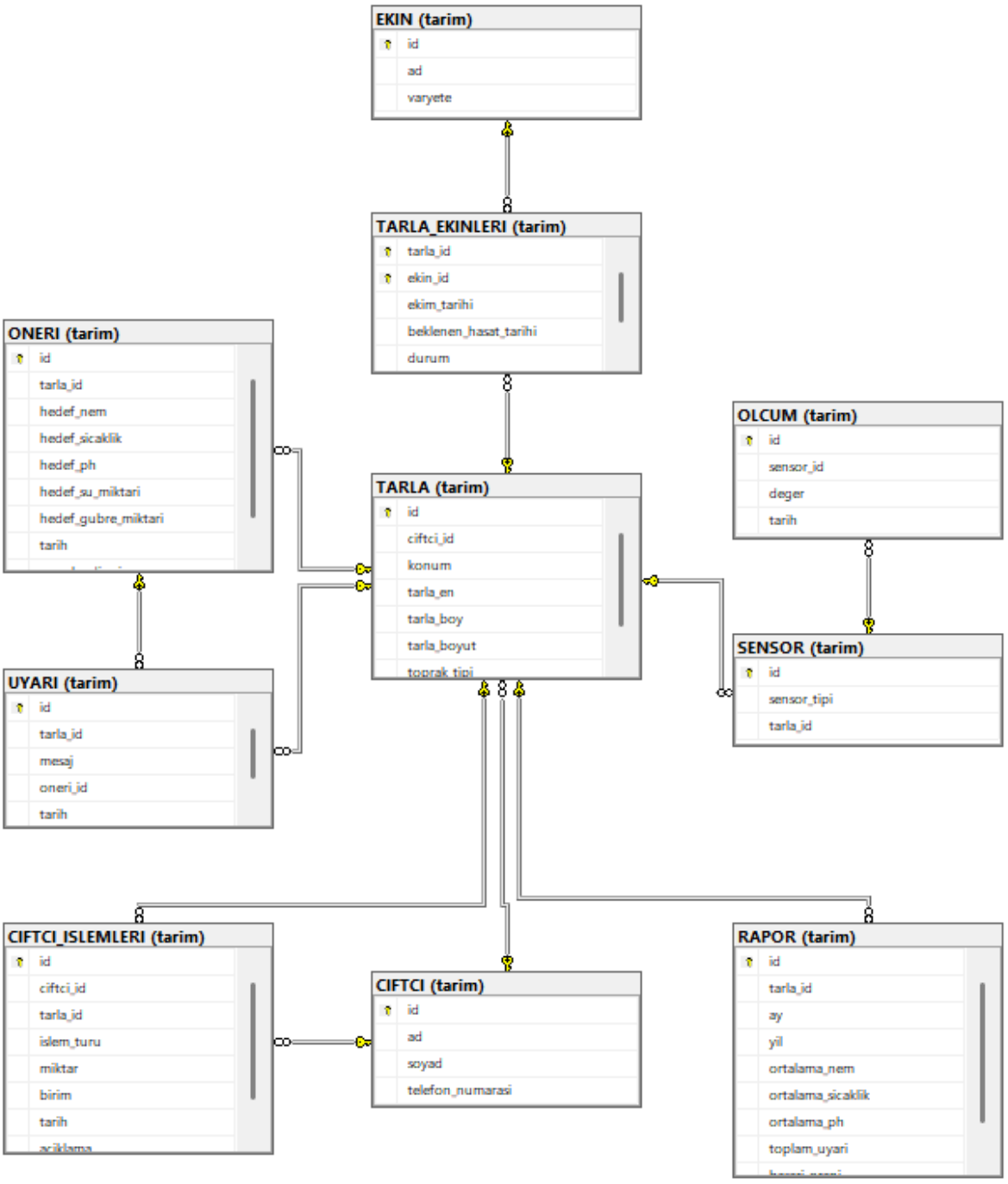
```
-- 1. CIFTCI Tablosu
CREATE TABLE tarim.CIFTCI (
    id INT PRIMARY KEY IDENTITY(1,1),
    ad NVARCHAR(50) NOT NULL,
    soyad NVARCHAR(50) NOT NULL,
    telefon_numarasi NVARCHAR(20)
);

-- 2. EKIN Tablosu
CREATE TABLE tarim.EKIN (
    id INT PRIMARY KEY IDENTITY(1,1),
    ad NVARCHAR(50) NOT NULL,
    varyete NVARCHAR(50)
);

-- 3. TARLA Tablosu
CREATE TABLE tarim.TARLA (
    id INT PRIMARY KEY IDENTITY(1,1),
    ciftci_id INT NOT NULL,
    konum NVARCHAR(100),
    tarla_en FLOAT,
    tarla_boy FLOAT,
    tarla_boyut FLOAT,
    toprak_tipi NVARCHAR(50),

    CONSTRAINT FK_Tarla_Ciftci FOREIGN KEY (ciftci_id) REFERENCES
tarim.CIFTCI(id)
);
```

```
-- 4. TARLA_EKINLERI (Junction Table)
CREATE TABLE tarim.TARLA_EKINLERI (
    tarla_id INT NOT NULL,
    ekin_id INT NOT NULL,
    ekim_tarihi DATE,
    beklenen_hasat_tarihi DATE,
    durum NVARCHAR(50),
    CONSTRAINT PK_TarlaEkinleri PRIMARY KEY (tarla_id, ekin_id),

    CONSTRAINT FK_TE_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id),
    CONSTRAINT FK_TE_Ekin FOREIGN KEY (ekin_id) REFERENCES
tarim.EKIN(id)
);

-- 5. SENSOR Tablosu
CREATE TABLE tarim.SENSOR (
    id INT PRIMARY KEY IDENTITY(1,1),
    sensor_tipi NVARCHAR(50),
    tarla_id INT NOT NULL,
    CONSTRAINT FK_Sensor_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 6. OLCUM Tablosu
CREATE TABLE tarim.OLCUM (
    id INT PRIMARY KEY IDENTITY(1,1),
    sensor_id INT NOT NULL,
    deger FLOAT NOT NULL,
    tarih DATETIME DEFAULT GETDATE(),
    CONSTRAINT FK_Olcum_Sensor FOREIGN KEY (sensor_id) REFERENCES
tarim.SENSOR(id)
);
```

```
-- 7. ONERI Tablosu
CREATE TABLE tarim.ONERI (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    hedef_nem FLOAT,
    hedef_sicaklik FLOAT,
    hedef_ph FLOAT,
    hedef_su_miktari FLOAT,
    hedef_gubre_miktari FLOAT,
    tarih DATETIME DEFAULT GETDATE(),
    uygulandi_mi BIT DEFAULT 0,
    CONSTRAINT FK_Oneri_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 8. UYARI Tablosu
CREATE TABLE tarim.UYARI (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    mesaj NVARCHAR(255),
    oneri_id INT NULL,
    tarih DATETIME DEFAULT GETDATE(),
    CONSTRAINT FK_Uyari_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id),
    CONSTRAINT FK_Uyari_Oneri FOREIGN KEY (oneri_id) REFERENCES
tarim.ONERI(id)
);

-- 9. RAPOR Tablosu
CREATE TABLE tarim.RAPOR (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
```

```
    ay INT,
    yil INT,
    ortalama_nem FLOAT,
    ortalama_sicaklik FLOAT,
    ortalama_ph FLOAT,
    toplam_uyari INT,
    basari_orani FLOAT,
    CONSTRAINT FK_Rapor_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 10. CIFTCI_ISLEMLERI Tablosu
CREATE TABLE tarim.CIFTCI_ISLEMLERI (
    id INT PRIMARY KEY IDENTITY(1,1),
    ciftci_id INT NOT NULL,
    tarla_id INT NOT NULL,
    islem_turu NVARCHAR(50),
    miktar FLOAT,
    birim NVARCHAR(20),
    tarih DATETIME DEFAULT GETDATE(),
    aciklama NVARCHAR(255),
    CONSTRAINT FK_Islem_Ciftci FOREIGN KEY (ciftci_id) REFERENCES
tarim.CIFTCI(id),
    CONSTRAINT FK_Islem_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);
```

6.3 Veri Tabanı Diyagramı



[VERİ TABANI DİYAGRAMI EKRAN GÖRÜNTÜSÜ]

6.4 Veri Tabanı Diyagramının Açıklaması

Akıllı Tarım ve Sulama Yönetim Sistemi için tasarlanan veri tabanı diyagramı; çiftçi, tarla, ekin, sensör ve bu varlıklardan üretilen ölçüm, öneri, uyarı ve rapor verilerinin bütüncül bir yapıda saklanmasını sağlayacak şekilde oluşturulmuştur. Veri tabanı, sistemin ihtiyaç duyduğu tüm operasyonel ve analitik süreçleri destekleyecek biçimde kurgulanmıştır.

Aşağıda veri tabanında yer alan tablolar ve bu tablolara ait alanlar açıklanmaktadır.

CIFTCI (Çiftçi)
Çiftçilere ait kişisel ve iletişim bilgilerini tutan tablodur.
Alanlar:
id (Primary Key)
ad
soyad
telefon_numarasi

EKIN (Ekin)
Tarım faaliyetlerinde yetiştirilen ürünlere ait bilgileri içerir.
Alanlar:
id (Primary Key)
ad
varyete

TARLA (Tarla)
Çiftçilere ait tarla bilgilerini tutar ve her tarla bir çiftçi ile ilişkilidir.
Alanlar:
id (Primary Key)
ciftci_id (Foreign Key – CIFTCI.id)
konum
tarla_en
tarla_boy
tarla_boyut
toprak_tipi

TARLA_EKINLERI (Tarla Ekinleri)
Tarlalar ile ekinler arasındaki çoktan çoğa ilişkiyi temsil eden ilişki tablosudur. Hangi

tarlada hangi ekinin ekildiği bu tablo üzerinden takip edilir.
Alanlar:
tarla_id (Primary Key, Foreign Key – TARLA.id)
ekin_id (Primary Key, Foreign Key – EKIN.id)
ekim_tarihi
beklenen_hasat_tarihi
durum

SENSOR (Sensör)
Tarlalara yerleştirilen sensörlere ait bilgileri tutar.
Alanlar:
id (Primary Key)
sensor_tipi
tarla_id (Foreign Key – TARLA.id)

OLCUM (Ölçüm)
Sensörlerden elde edilen ölçüm verilerinin kaydedildiği tablodur.
Alanlar:
id (Primary Key)
sensor_id (Foreign Key – SENSOR.id)
deger
tarih

ONERI (Öneri)
Sistem tarafından analizler sonucunda oluşturulan sulama ve tarımsal bakım önerilerini içerir.
Alanlar:
id (Primary Key)
tarla_id (Foreign Key – TARLA.id)
hedef_nem
hedef_sicaklik
hedef_ph
hedef_su_miktari
hedef_gubre_miktari
tarih
uygulandi_mi

UYARI (Uyarı)
Kritik eşiklerin aşılması durumunda oluşturulan sistem uyarılarını tutar.
Alanlar:
id (Primary Key)
tarla_id (Foreign Key – TARLA.id)
mesaj
oneri_id (Foreign Key – ONERI.id)
tarih

RAPOR (Rapor)
Tarlalara ait aylık ve yıllık performans özetlerini içeren rapor tablosudur.
Alanlar:
id (Primary Key)
tarla_id (Foreign Key – TARLA.id)
ay
yil
ortalama_nem
ortalama_sicaklik
ortalama_ph
toplam_uyari
basari_orani

CIFTCI_ISLEMLERI (Çiftçi İşlemleri)
Çiftçilerin tarlalar üzerinde gerçekleştirdiği tüm işlemlerin kayıt altına alındığı tablodur.
Alanlar:
id (Primary Key)
ciftci_id (Foreign Key – CIFTCI.id)
tarla_id (Foreign Key – TARLA.id)
islem_turu
miktar
birim
tarih
aciklama

7. ÖRNEK VERİLER

7.1 Veri Ekleme Komutları (INSERT)

-- CIFTCI Ekleme

INSERT INTO tarim.CIFTCI (ad, soyad, telefon_numarasi) VALUES

('Ahmet', 'Yılmaz', '05321112233'),

('Ayşe', 'Demir', '05442223344'),

('Mehmet', 'Öztürk', '05553334455');

-- EKIN Ekleme

INSERT INTO tarim.EKIN (ad, varyete) VALUES

('Buğday', 'Bezostaja'),

('Mısır', 'Hibrit'),

('Domates', 'Çeri');

-- TARLA Ekleme

INSERT INTO tarim.TARLA (ciftci_id, konum, tarla_en, tarla_boy, tarla_boyut, toprak_tipi) VALUES

(1, 'Konya Ovası - Parsel 1', 100, 200, 20000, 'Killi'),

(2, 'Adana Çukurova - Parsel 5', 50, 100, 5000, 'Tınlı'),

(3, 'Tekirdağ - Parsel 3', 80, 120, 9600, 'Kumlu');

-- TARLA_EKINLERI Ekleme

INSERT INTO tarim.TARLA_EKINLERI (tarla_id, ekin_id, ekim_tarihi, beklenen_hasat_tarihi, durum) VALUES

(1, 1, '2023-10-15', '2024-06-20', 'Büyüyor'),

(2, 2, '2024-04-01', '2024-09-15', 'Ekildi'),

(3, 3, '2024-05-10', '2024-08-30', 'Çiçeklenme');

-- SENSOR Ekleme

INSERT INTO tarim.SENSOR (sensor_tipi, tarla_id) VALUES

('Nem Sensörü', 1),

('Sıcaklık Sensörü', 2),

('PH Sensörü', 3);

-- OLCUM Ekleme

INSERT INTO tarim.OLCUM (sensor_id, deger, tarih) VALUES

(1, 45.5, '2024-05-20 10:00:00'),

(2, 28.3, '2024-05-20 10:05:00'),

(3, 6.8, '2024-05-20 10:10:00');

-- ONERI Ekleme

INSERT INTO tarim.ONERI (tarla_id, hedef_nem, hedef_sicaklik, hedef_ph, hedef_su_miktari, hedef_gubre_miktari, uygulandi_mi) VALUES

(1, 50.0, 25.0, 7.0, 1000, 50, 0),

(2, 60.0, 30.0, 6.5, 1500, 0, 1),

(3, 55.0, 24.0, 7.2, 800, 20, 0);

-- UYARI Ekleme

INSERT INTO tarim.UYARI (tarla_id, mesaj, oneri_id, tarih) VALUES

(1, 'Nem seviyesi kritik derecede düşük!', 1, '2024-05-21 09:00:00'),

(2, 'Sıcaklık normalin üzerinde, sulama gerekli.', NULL, '2024-05-21 12:30:00'),

(3, 'PH dengesi bozuldu, gübreleme kontrol edilmeli.', 3, '2024-05-21 15:45:00');

-- RAPOR Ekleme

INSERT INTO tarim.RAPOR (tarla_id, ay, yil, ortalama_nem, ortalama_sicaklik, ortalama_ph, toplam_uyari, basari_orani) VALUES

(1, 4, 2024, 48.5, 22.1, 7.1, 5, 85.5),

(2, 4, 2024, 55.2, 28.4, 6.4, 2, 92.0),

(3, 4, 2024, 50.0, 23.0, 7.0, 8, 76.4);

```
-- CIFTCI_ISLEMLERI Ekleme
INSERT INTO tarim.CIFTCI_ISLEMLERI (ciftci_id, tarla_id, islem_turu, miktar, birim, aciklama) VALUES
(1, 1, 'Sulama', 5000, 'Litre', 'Damlama sulama sistemi çalıştırıldı'),
(2, 2, 'Gübreleme', 100, 'Kg', 'Azotlu gübre takviyesi yapıldı'),
(3, 3, 'İlaçlama', 20, 'Litre', 'Böcek ilacı uygulandı');
```

8. SQL KOMUTLARI VE SCRIPT DOSYALARI

8.1 Gereksinim Bazlı SQL Komutları ve Script Dosyaları

[Her gereksinim için yazılan SQL komutlarını, script dosyalarını ve açıklamalarını yazınız.]

Veri tabanına çiftçi, ekin, tarla, tarla ekinleri, sensör, sensör değerleri, öneri, uyarı, rapor ve çiftçi işlemleri gibi varlıkları ve bu varlıkların özelliklerini de beraberinde eklememiz gerekiyordu. Bunu için de tüm bu işlemleri gerçekleştirebilmek adına tüm bu varlıklar için ayrı ayrı INSERT INTO komutunu kullanarak kod parçacıkları kullanarak kod parçacıkları hazırladık.

-- ÖRNEK VERİ GİRİŞİ

-- CIFTCI Ekleme

INSERT INTO tarim.CIFTCI (ad, soyad, telefon_numarasi) VALUES

('Ahmet', 'Yılmaz', '05321112233'),

('Ayşe', 'Demir', '05442223344'),

('Mehmet', 'Öztürk', '05553334455');

-- EKIN Ekleme

INSERT INTO tarim.EKIN (ad, varyete) VALUES

('Buğday', 'Bezostaja'),

('Mısır', 'Hibrit'),

('Domates', 'Çeri');

-- TARLA Ekleme

INSERT INTO tarim.TARLA (ciftci_id, konum, tarla_en, tarla_boy, tarla_boyut, toprak_tipi) VALUES

(1, 'Konya Ovası - Parsel 1', 100, 200, 20000, 'Killi'),

(2, 'Adana Çukurova - Parsel 5', 50, 100, 5000, 'Tınlı'),

(3, 'Tekirdağ - Parsel 3', 80, 120, 9600, 'Kumlu');

-- TARLA_EKINLERI Ekleme

```
INSERT INTO tarim.TARLA_EKINLERI (tarla_id, ekin_id, ekim_tarihi,
beklenen_hasat_tarihi, durum) VALUES
(1, 1, '2023-10-15', '2024-06-20', 'Büyüyor'),
(2, 2, '2024-04-01', '2024-09-15', 'Ekildi'),
(3, 3, '2024-05-10', '2024-08-30', 'Çiçeklenme');

-- SENSOR Ekleme
INSERT INTO tarim.SENSOR (sensor_tipi, tarla_id) VALUES
('Nem Sensörü', 1),
('Sıcaklık Sensörü', 2),
('PH Sensörü', 3);

-- OLCUM Ekleme
INSERT INTO tarim.OLCUM (sensor_id, deger, tarih) VALUES
(1, 45.5, '2024-05-20 10:00:00'),
(2, 28.3, '2024-05-20 10:05:00'),
(3, 6.8,  '2024-05-20 10:10:00');

-- ONERI Ekleme
INSERT INTO tarim.ONERI (tarla_id, hedef_nem, hedef_sicaklik, hedef_ph,
hedef_su_miktari, hedef_gubre_miktari, uygulandi_mi) VALUES
(1, 50.0, 25.0, 7.0, 1000, 50, 0),
(2, 60.0, 30.0, 6.5, 1500, 0, 1),
(3, 55.0, 24.0, 7.2, 800, 20, 0);

-- UYARI Ekleme
INSERT INTO tarim.UYARI (tarla_id, mesaj, oneri_id, tarih) VALUES
(1, 'Nem seviyesi kritik derecede düşük!', 1, '2024-05-21 09:00:00'),
(2, 'Sıcaklık normalin üzerinde, sulama gerekli.', NULL, '2024-05-21 12:30:00'),
(3, 'PH dengesi bozuldu, gübreleme kontrol edilmeli.', 3, '2024-05-21 15:45:00');

-- RAPOR Ekleme
```

```
INSERT INTO tarim.RAPOR (tarla_id, ay, yil, ortalama_nem, ortalama_sicaklik,
ortalama_ph, toplam_uyari, basari_orani) VALUES
(1, 4, 2024, 48.5, 22.1, 7.1, 5, 85.5),
(2, 4, 2024, 55.2, 28.4, 6.4, 2, 92.0),
(3, 4, 2024, 50.0, 23.0, 7.0, 8, 76.4);

-- CIFTCI_ISLEMLERI Ekleme
INSERT INTO tarim.CIFTCI_ISLEMLERI (ciftci_id, tarla_id, islem_turu, miktar, birim,
aciklama) VALUES
(1, 1, 'Sulama', 5000, 'Litre', 'Damlama sulama sistemi çalıştırıldı'),
(2, 2, 'Gübreleme', 100, 'Kg', 'Azotlu gübre takviyesi yapıldı'),
(3, 3, 'İlaçlama', 20, 'Litre', 'Böcek ilacı uygulandı');
```

Tabi bunun öncesinde bu verileri konumlandır bileceğimiz uygun alanlar için tabloları oluşturmamız gerekti, bunun içinde CREATE TABLE komutunu kullanarak her varlık için birer tablo oluşturduk, tabloların birincil ve ikincil (varsa) anahtarlarını belirledik.

```
USE Tarim_SistemiDB;
GO


--  TABLO OLUŞTURMA İŞLEMLERİ (Şema eklenmiş hali)


-- 1. CIFTCI Tablosu
CREATE TABLE tarim.CIFTCI (
    id INT PRIMARY KEY IDENTITY(1,1),
    ad NVARCHAR(50) NOT NULL,
    soyad NVARCHAR(50) NOT NULL,
    telefon_numarasi NVARCHAR(20)
);

-- 2. EKIN Tablosu
```

```
CREATE TABLE tarim.EKIN (
    id INT PRIMARY KEY IDENTITY(1,1),
    ad NVARCHAR(50) NOT NULL,
    varyete NVARCHAR(50)
);

-- 3. TARLA Tablosu
CREATE TABLE tarim.TARLA (
    id INT PRIMARY KEY IDENTITY(1,1),
    ciftci_id INT NOT NULL,
    konum NVARCHAR(100),
    tarla_en FLOAT,
    tarla_boy FLOAT,
    tarla_boyut FLOAT,
    toprak_tipi NVARCHAR(50),

    CONSTRAINT FK_Tarla_Ciftci FOREIGN KEY (ciftci_id) REFERENCES
tarim.CIFTCI(id)
);

-- 4. TARLA_EKINLERI (Junction Table)
CREATE TABLE tarim.TARLA_EKINLERI (
    tarla_id INT NOT NULL,
    ekin_id INT NOT NULL,
    ekim_tarihi DATE,
    beklenen_hasat_tarihi DATE,
    durum NVARCHAR(50),
    CONSTRAINT PK_TarlaEkinleri PRIMARY KEY (tarla_id, ekin_id),

    CONSTRAINT FK_TE_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id),
    CONSTRAINT FK_TE_Ekin FOREIGN KEY (ekin_id) REFERENCES
tarim.EKIN(id)
```

```
);

-- 5. SENSOR Tablosu
CREATE TABLE tarim.SENSOR (
    id INT PRIMARY KEY IDENTITY(1,1),
    sensor_tipi NVARCHAR(50),
    tarla_id INT NOT NULL,
    CONSTRAINT FK_Sensor_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 6. OLCUM Tablosu
CREATE TABLE tarim.OLCUM (
    id INT PRIMARY KEY IDENTITY(1,1),
    sensor_id INT NOT NULL,
    deger FLOAT NOT NULL,
    tarih DATETIME DEFAULT GETDATE(),
    CONSTRAINT FK_Olcum_Sensor FOREIGN KEY (sensor_id) REFERENCES
tarim.SENSOR(id)
);

-- 7. ONERI Tablosu
CREATE TABLE tarim.ONERI (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    hedef_nem FLOAT,
    hedef_sicaklik FLOAT,
    hedef_ph FLOAT,
    hedef_su_miktari FLOAT,
    hedef_gubre_miktari FLOAT,
    tarih DATETIME DEFAULT GETDATE(),
    uygulandi_mi BIT DEFAULT 0,
```

```
    CONSTRAINT FK_Oneri_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 8. UYARI Tablosu
CREATE TABLE tarim.UYARI (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    mesaj NVARCHAR(255),
    oneri_id INT NULL,
    tarih DATETIME DEFAULT GETDATE(),
    CONSTRAINT FK_Uyari_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id),
    CONSTRAINT FK_Uyari_Oneri FOREIGN KEY (oneri_id) REFERENCES
tarim.ONERI(id)
);

-- 9. RAPOR Tablosu
CREATE TABLE tarim.RAPOR (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    ay INT,
    yil INT,
    ortalama_nem FLOAT,
    ortalama_sicaklik FLOAT,
    ortalama_ph FLOAT,
    toplam_uyari INT,
    basari_orani FLOAT,
    CONSTRAINT FK_Rapor_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 10. CIFTCI_ISLEMLERI Tablosu
```

```
CREATE TABLE tarim.CIFTCI_ISLEMLERI (
    id INT PRIMARY KEY IDENTITY(1,1),
    ciftci_id INT NOT NULL,
    tarla_id INT NOT NULL,
    islem_turu NVARCHAR(50),
    miktar FLOAT,
    birim NVARCHAR(20),
    tarih DATETIME DEFAULT GETDATE(),
    aciklama NVARCHAR(255),
    CONSTRAINT FK_Islem_Ciftci FOREIGN KEY (ciftci_id) REFERENCES
tarim.CIFTCI(id),
    CONSTRAINT FK_Islem_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);
```

Kuraklık seviyesinin mahsüller için önemli olmasında ötürü su seviyesinin kritik seviyenin altına düşmesi halinde önlem almak adına AFTER INSERT trigger'ı tasarlanmıştır (kaynak ve test kodları ekran görüntüsü olarak ekte verilmiştir).

```
USE Tarim_SistemiDB;
GO

-- TRIGGER
IF OBJECT_ID('tarim.trg_KritikSeviyeUyari', 'TR') IS NOT NULL
   DROP TRIGGER tarim.trg_KritikSeviyeUyari;
GO

CREATE TRIGGER tarim.trg_KritikSeviyeUyari
ON tarim.OLCUM
AFTER INSERT
AS
BEGIN
```

```
    SET NOCOUNT ON;

    INSERT INTO tarim.UYARI (tarla_id, mesaj, tarih, oneri_id)
    SELECT
        s.tarla_id,
        CONCAT('KRİTİK UYARI: Toprak nem oranı çok düşük! (', i.deger, ')'),
        GETDATE(),
        NULL
    FROM
        INSERTED i
    INNER JOIN
        tarim.SENSOR s ON i.sensor_id = s.id
    WHERE
        s.sensor_tipi = 'Nem'
        AND i.deger < 30.0;

    IF @@ROWCOUNT > 0
    BEGIN
        PRINT 'DİKKAT: Toplu veri girişi algılandı ve gerekli uyarılar oluşturuldu.';
    END
END;
GO

-- STORED PROCEDURE: Ölçüm Ekleme
CREATE PROCEDURE tarim.sp_OlcumEkle
    @SensorId INT,
    @Deger FLOAT,
    @Tarih DATETIME
AS
BEGIN
    SET NOCOUNT ON;
```

```
    INSERT INTO tarim.OLCUM (sensor_id, deger, tarih)
    VALUES (@SensorId, @Deger, @Tarih);

    PRINT 'Ölçüm başarıyla eklendi.';
END;
GO

-- STORED PROCEDURE: Atomik İşlem (Ekim)
CREATE PROCEDURE tarim.sp_EkinEkimIslemi
    @CiftciId INT,
    @TarlaId INT,
    @EkinId INT,
    @EkimTarihi DATE,
    @Aciklama NVARCHAR(MAX)
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRANSACTION;

    BEGIN TRY
        INSERT INTO tarim.TARLA_EKINLERI (tarla_id, ekin_id, ekim_tarihi, durum)
        VALUES (@TarlaId, @EkinId, @EkimTarihi, 'Ekildi');

        INSERT INTO tarim.CIFTCI_ISLEMLERI (ciftci_id, tarla_id, islem_turu, miktar,
birim, tarih, aciklama)
        VALUES (@CiftciId, @TarlaId, 'Ekim', 1, 'Dönüm', GETDATE(), @Aciklama);

        COMMIT TRANSACTION;
        PRINT 'İşlem Başarılı: Ekin ekildi ve kayıt günlüğe işlendi.';
    END TRY
    BEGIN CATCH
```

```
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
        PRINT 'HATA OLUŞTU! İşlem iptal edildi.';
        PRINT 'Hata Detayı: ' + @ErrorMessage;
    END CATCH
END;
GO
```

9. SAKLI YORDAM VE TETİKLEYİCİ

9.1 Saklı Yordam Adı (Stored Procedure)

9.1.1 Amaç ve İş Operasyonu

Bu saklı yordam (sp_OlcumEkle), sensörlerden gelen sıcaklık ve nem verilerini OLCUM tablosuna standart bir şekilde kaydetmek için oluşturulmuştur. İşlem sırasında SET NOCOUNT ON komutu kullanılarak ağ trafiği azaltılır ve veri giriş performansı artırılır.

9.1.2 Saklı Yordam Kodu

```
CREATE PROCEDURE tarim.sp_OlcumEkle

@SensorId INT,

@Deger FLOAT,

@Tarih DATETIME

AS

BEGIN

SET NOCOUNT ON; -- Network trafiğini azaltmak için performans ayarı

-- Hata yönetimi olmadan basit ekleme (Trigger devreye girecek)
```

```
INSERT INTO tarim.OLCUM (sensor_id, deger, tarih)
VALUES (@SensorId, @Deger, @Tarih);
PRINT 'Ölçüm başarıyla eklendi.';
END;
GO
```

9.1.3 Test Senaryoları ve Sonuçları

Ölçüm başarıyla eklendi.
Ölçüm başarıyla eklendi.

[Her tetikleyiciyi aşağıda verilen 4 madde şeklinde açıklayınız.]

9.2 Tetikleyici (Trigger)

9.2.1 Amaç ve Tetikleme Koşulu

Bu tetikleyici (trg_KritikSeviyeUyari), OLCUM tablosuna yeni bir veri eklendiğinde (AFTER INSERT) otomatik olarak devreye girer. Eğer eklenen verinin sensör tipi 'Nem' ise ve değeri 30.0'ın altındaysa, sistem UYARI tablosuna otomatik olarak kritik seviye uyarısı kaydeder

9.2.2 Tetikleyici Kodu

```
USE Tarim_SistemiDB;
GO

-- Eğer trigger zaten varsa sil, yeniden oluştur (Hata almamak için)
IF OBJECT_ID('tarim.trg_KritikSeviyeUyari', 'TR') IS NOT NULL
   DROP TRIGGER tarim.trg_KritikSeviyeUyari;
GO

CREATE TRIGGER tarim.trg_KritikSeviyeUyari
ON tarim.OLCUM
AFTER INSERT
AS
```

```
BEGIN
    SET NOCOUNT ON;

    -- Mantık: Eklenen ölçüm 'Nem' tipindeyse ve değeri 30'un altındaysa UYARI
tablosuna yaz.
    INSERT INTO tarim.UYARI (tarla_id, mesaj, tarih, oneri_id)
    SELECT
        s.tarla_id,
        CONCAT('KRİTİK UYARI: Toprak nem oranı çok düşük! (', i.deger, ')'),
        GETDATE(),
        NULL
    FROM
        INSERTED i
    INNER JOIN
        tarim.SENSOR s ON i.sensor_id = s.id
    WHERE
        s.sensor_tipi = 'Nem'   -- Sadece Nem sensörleri
        AND i.deger < 30.0;     -- Kritik eşik değeri (Testindeki 20 bunu tetikler)

    -- Konsola bilgi mesajı bas (Opsiyonel)
    IF @@ROWCOUNT > 0
    BEGIN
        PRINT '>> DİKKAT: Kritik seviye tespit edildi, UYARI tablosuna kayıt eklendi.';
    END
END;
GO
```

9.2.3 Test Senaryoları ve Sonuçları

- 1. TRİGGER TEST (Otomatik Uyarı Sistemi)

```
-- ========================================
```

-- Veri Taban ndaki Durum: Sensor ID 1, "Nem Sensörüdürr ve Ahmet'in tarlasına (Tarla İD: 1) aittir.

```
PRINT '--- TRİGGER TESTİ BAŞLIYOR ---';


-- Senaryo A: Normal Deger
-- Ahmet'in tarlasına %55 nem değeri giriliyor.
EXEC tarim.sp_OlcumEkle
  @SensorId = 1,      -- Mevcut Nem Sensörü
  @Deger = 55.0,      -- Güvenli Değer
  @Tarih = '2024-06-01 10:00:00';


-- Senaryo B: Kritik Değer
-- Ahmet'in tarlas na kritik seviyede (%20) nem değeri giriliyor.
-- Beklenen: Trigger  al  mal  ve UYARI tablosuna kayıt atma .
EXEC tarim.sp_OlcumEkle
  @SensorId = 1,      -- Mevcut Nem Sensörü
  @Deger = 20.0,      -- Kritik De er (<30)
  @Tarih = '2024-06-01 12:00:00';


-- KONTROL
PRINT 'Trigger Sonu lar :';
SELECT * FROM tarim.UYARI WHERE tarla_id = 1;
GO
```

-- TRANSACTION TESTİ BAŞLIYOR ---

İşlem Başarılı: Ekin ekildi ve kayıt günlüğe işlendi.

Başarılı İşlem Kontrolü:

(1 satır etkilendi)

(1 satır etkilendi)

10. TRANSACTION YÖNETİMİ

10.1 Transaction Senaryosu Adı

Ekim İşlemi Atomikliği: Bir çiftçi tarlasına ekin ektiğinde, sistemde iki işlem yapılmalıdır:
1) TARLA_EKINLERI tablosuna ekim kaydı girilmeli.
2) CIFTCI_ISLEMLERI tablosuna günlük logu düşülmelidir. Bu senaryoda BEGIN TRANSACTION kullanılarak bu iki işlemin bir bütün (atomik) olması sağlanır. Herhangi bir adımda hata olursa tüm işlemler ROLLBACK ile geri alınır.

10.2 Transaction Kodu

```
-- STORED PROCEDURE: Ölçüm Ekleme
CREATE PROCEDURE tarim.sp_OlcumEkle
    @SensorId INT,
    @Deger FLOAT,
    @Tarih DATETIME
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO tarim.OLCUM (sensor_id, deger, tarih)
    VALUES (@SensorId, @Deger, @Tarih);

    PRINT 'Ölçüm başarıyla eklendi.';
END;
GO

-- STORED PROCEDURE: Atomik İşlem (Ekim)
CREATE PROCEDURE tarim.sp_EkinEkimIslemi
    @CiftciId INT,
    @TarlaId INT,
    @EkinId INT,
    @EkimTarihi DATE,
    @Aciklama NVARCHAR(MAX)
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRANSACTION;

    BEGIN TRY
        INSERT INTO tarim.TARLA_EKINLERI (tarla_id, ekin_id, ekim_tarihi, durum)
```

```
        VALUES (@TarlaId, @EkinId, @EkimTarihi, 'Ekildi');

        INSERT INTO tarim.CIFTCI_ISLEMLERI (ciftci_id, tarla_id, islem_turu, miktar,
birim, tarih, aciklama)
        VALUES (@CiftciId, @TarlaId, 'Ekim', 1, 'Dönüm', GETDATE(), @Aciklama);

        COMMIT TRANSACTION;
        PRINT 'İşlem Başarılı: Ekin ekildi ve kayıt günlüğe işlendi.';
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
        PRINT 'HATA OLUŞTU! İşlem iptal edildi.';
        PRINT 'Hata Detayı: ' + @ErrorMessage;
    END CATCH
END;
GO
```

10.3 Başarılı Senaryo Testi

```
-- TRANSACTION TESTİ BAŞLIYOR ---
İşlem Başarılı: Ekin ekildi ve kayıt günlüğe işlendi.
Başarılı İşlem Kontrolü:
(1 satır etkilendi)

(1 satır etkilendi)
```

10.4 Hata Senaryosu ve ROLLBACK Testi

--- ROLLBACK TESTİ BAŞLIYOR (Kırmızı Hata Mesajı Normaldir) ---

HATA OLUŞTU! İşlem iptal edildi.

Hata Detayı: The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Islem_Ciftci". The conflict occurred in database "Tarim_SistemiDB", table "tarim.CIFTCI", column 'id'.

Rollback Sonrası Kontrol (Sonuç Boş Gelmeli):

(0 satır etkilendi)

11. TAKIM ÇALIŞMASI VE GÖREV DAĞILIMI

11.1 Takım Üyeleri ve Rolleri

Takım Üyesi	Öğrenci No	Temel Rol
Emircan ÖZKAYA	235260029	Proje Lideri
Ali Kağan AKDEMİR	235260035	[SQL Geliştirici / Test Sorumlusu]
Taha Buğra ÇİÇEK	235260083	[Dokümantasyon / Kalite Kontrol]

11.2 Gerçekleştirilen İşler ve Sorumluluk Matrisi

[Aşağıdaki tabloda projede gerçekleştirilen her iş için talimat veren, gerçekleştiren ve kontrol eden kişiler belirtilecektir.]

No	Yapılan İş / Özellik	Talimat Veren	Gerçekleştiren	Kontrol Eden	Durum
1	Proje gereksinimlerinin belirlenmesi	Emircan ÖZKAYA	Taha Buğra ÇİÇEK	Ali Kağan AKDEMİR	✓
2	E-R diyagramının oluşturulması	Taha Buğra ÇİÇEK	Ali Kağan AKDEMİR	Emircan ÖZKAYA	✓
3	İlişkisel şemalara dönüştürme	Taha Buğra ÇİÇEK	Emircan ÖZKAYA	Ali Kağan AKDEMİR	✓
4	Normalizasyon (3NF/BCNF)	Ali Kağan AKDEMİR	Emircan ÖZKAYA	Taha Buğra ÇİÇEK	✓
5	SQL Server'da şema oluşturma	Ali Kağan AKDEMİR	Emircan ÖZKAYA	Taha Buğra ÇİÇEK	✓
6	Birincil/Yabancı anahtar tanımları	Taha Buğra ÇİÇEK	Ali Kağan AKDEMİR	Emircan ÖZKAYA	✓
7	Nitelik kısıtlamalarının eklenmesi	Emircan ÖZKAYA	Taha Buğra ÇİÇEK	Ali Kağan AKDEMİR	✓
8	Örnek verilerin eklenmesi	Emircan ÖZKAYA	Taha Buğra ÇİÇEK	Ali Kağan AKDEMİR	✓
9	SQL sorgu script dosyalarının hazırlanması	Ali Kağan AKDEMİR	Emircan ÖZKAYA	Taha Buğra ÇİÇEK	✓

10	Saklı yordamın (Stored Procedure) yazılması	Emircan Özkaya	Ali Kağan AKDEMİR	Taha Buğra ÇİÇEK	✓
11	Tetikleyicinin (Trigger) yazılması	Emircan Özkaya	Ali Kağan AKDEMİR	Taha Buğra ÇİÇEK	✓
12	Transaction yönetimi kodunun yazılması	Emircan Özkaya	Ali Kağan AKDEMİR	Taha Buğra ÇİÇEK	✓
13	COMMIT/ROLLBACK test senaryoları	Ali Kağan AKDEMİR	Emircan Özkaya	Taha Buğra ÇİÇEK	✓
14	Rapor hazırlama ve dokümantasyon	Ali Kağan AKDEMİR	Taha Buğra ÇİÇEK	Emircan ÖZKAYA	✓
15	Son kontrol ve teslim	Emircan ÖZKAYA	Taha Buğra ÇİÇEK	Ali Kağan AKDEMİR	✓

Not: Talimat Veren: İlgili işin yapılması gerektiğine karar veren ve yönlendiren kişi. Gerçekleştiren: İşi fiilen yapan kişi. Kontrol Eden: İşin doğruluğunu ve kalitesini kontrol eden kişi.

12. SONUÇ VE DEĞERLENDİRME

Kod kısmında çok büyük bir tecrübe edinmedik projenin genel hatları doğrultusunda, ancak veri tabanının sadece verileri bir depolama alanı olmadığını aynı zamanda içerisinde gerekti noktalara ilişkin hata yönetimin olması gerektiğini gördük ve öneminin farkına vardık.
Öte yandan varlıkları, özellikleri ve bu varlıklar arasındaki ilişkileri tasarlarken hatalara düştük, vakit ayırıp üzerinde düşünmemiz gerekti. Tabi bu süreç ilerleyen zamanlarda daha komplike veri tabanı sistemleri hazırlaya bilmemiz adına bir alt yapı niteliğinde oldu bizim için.

EKLER

EK-A: SQL Script Dosyaları

https://drive.google.com/drive/folders/1bXB_bCLMh5DwZwoXZ5c--SsOp1WKcLo9?usp=sharing

EK-B:

Tablo Oluşturma Komutları (SQL Scriptleri)

```
-- 1. CIFTCI Tablosu
CREATE TABLE tarim.CIFTCI (
    id INT PRIMARY KEY IDENTITY(1,1),
    ad NVARCHAR(50) NOT NULL,
    soyad NVARCHAR(50) NOT NULL,
    telefon_numarasi NVARCHAR(20)
);

-- 2. EKIN Tablosu
CREATE TABLE tarim.EKIN (
    id INT PRIMARY KEY IDENTITY(1,1),
    ad NVARCHAR(50) NOT NULL,
    varyete NVARCHAR(50)
);

-- 3. TARLA Tablosu
CREATE TABLE tarim.TARLA (
    id INT PRIMARY KEY IDENTITY(1,1),
    ciftci_id INT NOT NULL,
    konum NVARCHAR(100),
    tarla_en FLOAT,
    tarla_boy FLOAT,
    tarla_boyut FLOAT,
    toprak_tipi NVARCHAR(50),
```

```
    CONSTRAINT FK_Tarla_Ciftci FOREIGN KEY (ciftci_id) REFERENCES
tarim.CIFTCI(id)
);

-- 4. TARLA_EKINLERI (Junction Table)
CREATE TABLE tarim.TARLA_EKINLERI (
    tarla_id INT NOT NULL,
    ekin_id INT NOT NULL,
    ekim_tarihi DATE,
    beklenen_hasat_tarihi DATE,
    durum NVARCHAR(50),
    CONSTRAINT PK_TarlaEkinleri PRIMARY KEY (tarla_id, ekin_id),
    CONSTRAINT FK_TE_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id),
    CONSTRAINT FK_TE_Ekin FOREIGN KEY (ekin_id) REFERENCES
tarim.EKIN(id)
);

-- 5. SENSOR Tablosu
CREATE TABLE tarim.SENSOR (
    id INT PRIMARY KEY IDENTITY(1,1),
    sensor_tipi NVARCHAR(50),
    tarla_id INT NOT NULL,
    CONSTRAINT FK_Sensor_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 6. OLCUM Tablosu
CREATE TABLE tarim.OLCUM (
    id INT PRIMARY KEY IDENTITY(1,1),
    sensor_id INT NOT NULL,
    deger FLOAT NOT NULL,
    tarih DATETIME DEFAULT GETDATE(),
```

```
    CONSTRAINT FK_Olcum_Sensor FOREIGN KEY (sensor_id) REFERENCES
tarim.SENSOR(id)
);

-- 7. ONERI Tablosu
CREATE TABLE tarim.ONERI (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    hedef_nem FLOAT,
    hedef_sicaklik FLOAT,
    hedef_ph FLOAT,
    hedef_su_miktari FLOAT,
    hedef_gubre_miktari FLOAT,
    tarih DATETIME DEFAULT GETDATE(),
    uygulandi_mi BIT DEFAULT 0,
    CONSTRAINT FK_Oneri_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 8. UYARI Tablosu
CREATE TABLE tarim.UYARI (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    mesaj NVARCHAR(255),
    oneri_id INT NULL,
    tarih DATETIME DEFAULT GETDATE(),
    CONSTRAINT FK_Uyari_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id),
    CONSTRAINT FK_Uyari_Oneri FOREIGN KEY (oneri_id) REFERENCES
tarim.ONERI(id)
);

-- 9. RAPOR Tablosu
```

```
CREATE TABLE tarim.RAPOR (
    id INT PRIMARY KEY IDENTITY(1,1),
    tarla_id INT NOT NULL,
    ay INT,
    yil INT,
    ortalama_nem FLOAT,
    ortalama_sicaklik FLOAT,
    ortalama_ph FLOAT,
    toplam_uyari INT,
    basari_orani FLOAT,
    CONSTRAINT FK_Rapor_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);

-- 10. CIFTCI_ISLEMLERI Tablosu
CREATE TABLE tarim.CIFTCI_ISLEMLERI (
    id INT PRIMARY KEY IDENTITY(1,1),
    ciftci_id INT NOT NULL,
    tarla_id INT NOT NULL,
    islem_turu NVARCHAR(50),
    miktar FLOAT,
    birim NVARCHAR(20),
    tarih DATETIME DEFAULT GETDATE(),
    aciklama NVARCHAR(255),
    CONSTRAINT FK_Islem_Ciftci FOREIGN KEY (ciftci_id) REFERENCES
tarim.CIFTCI(id),
    CONSTRAINT FK_Islem_Tarla FOREIGN KEY (tarla_id) REFERENCES
tarim.TARLA(id)
);
```

Veri Ekleme Komutları (INSERT)

```
-- CIFTCI Ekleme
```

```
INSERT INTO tarim.CIFTCI (ad, soyad, telefon_numarasi) VALUES
('Ahmet', 'Yılmaz', '05321112233'),
('Ayşe', 'Demir', '05442223344'),
('Mehmet', 'Öztürk', '05553334455');

-- EKIN Ekleme
INSERT INTO tarim.EKIN (ad, varyete) VALUES
('Buğday', 'Bezostaja'),
('Mısır', 'Hibrit'),
('Domates', 'Çeri');

-- TARLA Ekleme
INSERT INTO tarim.TARLA (ciftci_id, konum, tarla_en, tarla_boy, tarla_boyut,
toprak_tipi) VALUES
(1, 'Konya Ovası - Parsel 1', 100, 200, 20000, 'Killi'),
(2, 'Adana Çukurova - Parsel 5', 50, 100, 5000, 'Tınlı'),
(3, 'Tekirdağ - Parsel 3', 80, 120, 9600, 'Kumlu');

-- TARLA_EKINLERI Ekleme
INSERT INTO tarim.TARLA_EKINLERI (tarla_id, ekin_id, ekim_tarihi,
beklenen_hasat_tarihi, durum) VALUES
(1, 1, '2023-10-15', '2024-06-20', 'Büyüyor'),
(2, 2, '2024-04-01', '2024-09-15', 'Ekildi'),
(3, 3, '2024-05-10', '2024-08-30', 'Çiçeklenme');

-- SENSOR Ekleme
INSERT INTO tarim.SENSOR (sensor_tipi, tarla_id) VALUES
('Nem Sensörü', 1),
('Sıcaklık Sensörü', 2),
('PH Sensörü', 3);

-- OLCUM Ekleme
```

```
INSERT INTO tarim.OLCUM (sensor_id, deger, tarih) VALUES
(1, 45.5, '2024-05-20 10:00:00'),
(2, 28.3, '2024-05-20 10:05:00'),
(3, 6.8, '2024-05-20 10:10:00');

-- ONERI Ekleme
INSERT INTO tarim.ONERI (tarla_id, hedef_nem, hedef_sicaklik, hedef_ph,
hedef_su_miktari, hedef_gubre_miktari, uygulandi_mi) VALUES
(1, 50.0, 25.0, 7.0, 1000, 50, 0),
(2, 60.0, 30.0, 6.5, 1500, 0, 1),
(3, 55.0, 24.0, 7.2, 800, 20, 0);

-- UYARI Ekleme
INSERT INTO tarim.UYARI (tarla_id, mesaj, oneri_id, tarih) VALUES
(1, 'Nem seviyesi kritik derecede düşük!', 1, '2024-05-21 09:00:00'),
(2, 'Sıcaklık normalin üzerinde, sulama gerekli.', NULL, '2024-05-21 12:30:00'),
(3, 'PH dengesi bozuldu, gübreleme kontrol edilmeli.', 3, '2024-05-21 15:45:00');

-- RAPOR Ekleme
INSERT INTO tarim.RAPOR (tarla_id, ay, yil, ortalama_nem, ortalama_sicaklik,
ortalama_ph, toplam_uyari, basari_orani) VALUES
(1, 4, 2024, 48.5, 22.1, 7.1, 5, 85.5),
(2, 4, 2024, 55.2, 28.4, 6.4, 2, 92.0),
(3, 4, 2024, 50.0, 23.0, 7.0, 8, 76.4);

-- CIFTCI_ISLEMLERI Ekleme
INSERT INTO tarim.CIFTCI_ISLEMLERI (ciftci_id, tarla_id, islem_turu, miktar, birim,
aciklama) VALUES
(1, 1, 'Sulama', 5000, 'Litre', 'Damlama sulama sistemi çalıştırıldı'),
(2, 2, 'Gübreleme', 100, 'Kg', 'Azotlu gübre takviyesi yapıldı'),
(3, 3, 'İlaçlama', 20, 'Litre', 'Böcek ilacı uygulandı');
```

Test Senaryoları Kodları (Trigger ve Transaction)

```
-- 1. TRIGGER TESTİ (Otomatik Uyarı Sistemi)
-- Senaryo B: Kritik Değer Girişi
EXEC tarim.sp_OlcumEkle
 @SensorId = 1,
 @Deger = 20.0, -- Kritik Değer (<30)
 @Tarih = '2024-06-01 12:00:00';

-- KONTROL: UYARI tablosuna kayıt düştü mü?
SELECT * FROM tarim.UYARI WHERE tarla_id = 1;

-- 2. TRANSACTION (ATOMICITY) TESTİ
-- Senaryo A: Başarılı İşlem (COMMIT)
EXEC tarim.sp_EkinEkimIslemi
 @CiftciId = 1,
 @TarlaId = 1,
 @EkinId = 3,
 @EkimTarihi = '2024-06-10',
 @Aciklama = 'Ahmet Bey tarlasına Domates ekti. (Başarılı İşlem)';

-- KONTROL (Başarılı)
SELECT * FROM tarim.TARLA_EKINLERI WHERE tarla_id = 1 AND ekin_id = 3;
SELECT * FROM tarim.CIFTCI_ISLEMLERI WHERE aciklama LIKE '%Domates
ekti%';

-- Senaryo B: Hatalı İşlem (ROLLBACK) - Olmayan Çiftçi ID'si
EXEC tarim.sp_EkinEkimIslemi
 @CiftciId = 9999, -- HATA KAYNAĞI
 @TarlaId = 1,
 @EkinId = 2,
 @EkimTarihi = '2024-06-10',
```

```
 @Aciklama = 'Bu işlem geri alınmalı!';

-- KONTROL (Rollback Başarılı mı?) - Kayıt eklenmemiş olmalı
SELECT * FROM tarim.TARLA_EKINLERI WHERE tarla_id = 1 AND ekin_id = 2;
```

Saklı Yordam ve Trigger Tanımları

```
-- Trigger: Kritik Seviye Uyarısı
CREATE TRIGGER tarim.trg_KritikSeviyeUyari
ON tarim.OLCUM
AFTER INSERT
AS
BEGIN
 SET NOCOUNT ON;
 INSERT INTO tarim.UYARI (tarla_id, mesaj, tarih, oneri_id)
 SELECT s.tarla_id, CONCAT('KRİTİK UYARI: Toprak nem oranı çok düşük! (', i.deger, ')'), GETDATE(), NULL
 FROM INSERTED i
 INNER JOIN tarim.SENSOR s ON i.sensor_id = s.id
 WHERE s.sensor_tipi = 'Nem' AND i.deger < 30.0;
END;

-- Stored Procedure: Ölçüm Ekleme
CREATE PROCEDURE tarim.sp_OlcumEkle
 @SensorId INT, @Deger FLOAT, @Tarih DATETIME
AS
BEGIN
 INSERT INTO tarim.OLCUM (sensor_id, deger, tarih) VALUES (@SensorId, @Deger, @Tarih);
 PRINT 'Ölçüm başarıyla eklendi.';
END;
```

```
-- Stored Procedure: Atomik İşlem (Ekim)
CREATE PROCEDURE tarim.sp_EkinEkimIslemi
 @CiftciId INT, @TarlaId INT, @EkinId INT, @EkimTarihi DATE, @Aciklama
NVARCHAR(MAX)
AS
BEGIN
 BEGIN TRANSACTION;
 BEGIN TRY
  INSERT INTO tarim.TARLA_EKINLERI (tarla_id, ekin_id, ekim_tarihi, durum)
  VALUES (@TarlaId, @EkinId, @EkimTarihi, 'Ekildi');

  INSERT INTO tarim.CIFTCI_ISLEMLERI (ciftci_id, tarla_id, islem_turu, miktar,
birim, tarih, aciklama)
  VALUES (@CiftciId, @TarlaId, 'Ekim', 1, 'Dönüm', GETDATE(), @Aciklama);

  COMMIT TRANSACTION;
  PRINT 'İşlem Başarılı: Ekin ekildi ve kayıt günlüğe işlendi.';
 END TRY
 BEGIN CATCH
  ROLLBACK TRANSACTION;
  PRINT 'HATA OLUŞTU! İşlem iptal edildi.';
 END CATCH
END;
```

EK-C: Test Sonuçları

--- TRIGGER TESTİ BAŞLIYOR ---

Ölçüm başarıyla eklendi.

Ölçüm başarıyla eklendi.

Trigger Sonuçları:

(1 satır etkilendi)

--- TRANSACTION TESTİ BAŞLIYOR ---

İşlem Başarılı: Ekin ekildi ve kayıt günlüğe işlendi.

Başarılı İşlem Kontrolü:

(1 satır etkilendi)

(1 satır etkilendi)

--- ROLLBACK TESTİ BAŞLIYOR (Kırmızı Hata Mesajı Normaldir) ---

HATA OLUŞTU! İşlem iptal edildi.

Hata Detayı: The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Islem_Ciftci". The conflict occurred in database "Tarim_SistemiDB", table "tarim.CIFTCI", column 'id'.

Rollback Sonrası Kontrol (Sonuç Boş Gelmeli):

(0 satır etkilendi)

Tamamlanma süresi: 2025-12-20T16:34:20.8673276+03:00

Kayıt Tipi	**ID / Tarla ID**	**İlgili Bilgi / Mesaj**	**Tarih**	**Durum / Sonuç**
Kritik Uyarı (Trigger)	1	Nem seviyesi kritik derecede düşük!	2024-05-21 09:00	Sistem Otomatik Uyarı Oluşturdu
Ekin Durumu (Trans.)	1	Ekin ID: 3 (Domates)	2024-06-05	Başarıyla Güncellendi (Ekildi)
İşlem Logu (Trans.)	1	1 Dönüm Domates Ekimi (Çiftçi ID: 4)	2025-12-20 16:34	İşlem Başarıyla Kaydedildi
Hatalı İşlem (Rollback)	-	(Hata nedeniyle veri yazılmadı)	-	**İşlem Geri Alındı (Boş)**