BMÜ329 Veri Tabanı Sistemleri Dersi Dönem Projesi

Proje Başlığı: Öğrenci Bilgi Sistemi (OBS)

Proje Ekibindeki Kişiler:

- 220260006 Mehmet Miraç Özmen
- 220260018 Alperen Aktaş
- 220260028 Yunus Emre Gümüş

Öğrenci Bilgi Sistemi (OBS) Veri Tabanı Gereksinimleri

1. Proje Kullanıcıları Gereksinimleri ve Yetkileri

Öğrenci Bilgi Sistemi'nde üç ana kullanıcı türü bulunur: Öğrenciler, Öğretmenler ve Yöneticiler. Her kullanıcı türü, veri tabanındaki verilere farklı düzeylerde erişim ve müdahale yetkisine sahiptir.

1.1 Öğrenciler:

- Kendi tablolarını görüntüleyebilir.
- Devamsızlık kayıtlarını ve sınav sonuçlarını görebilir.
- Ders programına ve sınav takvimine erişebilir.
- Yalnızca okuma yetkisine sahiptir, veri girişi yapamaz.

1.2 Öğretmenler:

- Kendi verdikleri derslerin öğrenci listelerine, devamsızlık kayıtlarına ve sınav sonuçlarına erişebilirler.
- Dersleri için sınav oluşturabilir ve sınav sonuçlarını güncelleyebilirler.
- Öğrencilerin ders içindeki performans durumunu görebilirler.
- Hem okuma hem yazma yetkisine sahiptir.

1.3 Yöneticiler:

- Tüm öğrenci ve öğretmen kayıtlarını yönetme yetkisine sahiptir.
- Yeni dersler oluşturabilir, ders programlarını ve sınavları düzenleyebilir.
- Öğrencilerin ve öğretmenlerin bilgilerini güncelleyebilir veya sistemden çıkarabilirler.
- Veri tabanındaki tüm verilere tam erişim (okuma, yazma, güncelleme ve silme) yetkisine sahiptir.

2. Tablolar ve Varlıkların Özellikleri

Her tablo, veri tabanındaki bir varlık grubunu temsil eder. Bu varlıkların özellikleri (nitelikler), varlıklar arasındaki ilişkiler ve her ilişkideki sayısal kısıtlamalar aşağıda açıklanmıştır:

2.1 Öğrenciler (Students):

- **ID:** Birincil anahtar (PK), benzersiz öğrenci kimliği.
- **Department ID:** Öğrencinin bağlı olduğu bölüm (FK, Departments tablosuna bağlı).
- **Advisor ID:** Öğrencinin bağlı olduğu danışman (FK, Teachers tablosuna bağlı).
- **Role ID:** Öğrencinin bağlı olduğu yetki (FK, Roles tablosuna bağlı).
- **Student No:** Benzersiz öğrenci numarası, her öğrenci için tekil ve zorunlu.
- **Name:** Öğrenci adı, zorunlu bir metin alanı.
- **Surname:** Öğrenci soyadı, zorunlu bir metin alanı.

2.2 Öğretmenler (Teachers):

- **ID:** Birincil anahtar (PK), benzersiz öğretmen kimliği.
- **Department ID:** Öğretmenin bağlı olduğu bölüm (FK, Departments tablosuna bağlı).
- **Role ID:** Öğretmenin bağlı olduğu yetki (FK, Roles tablosuna bağlı).
- **Name:** Öğretmen adı, zorunlu bir metin alanı.
- **Surname:** Öğretmen soyadı, zorunlu bir metin alanı.

2.3 Yöneticiler (Admins):

- **ID:** Birincil anahtar (PK), benzersiz yönetici kimliği.
- **Role ID:** Yöneticinin bağlı olduğu yetki (FK, Roles tablosuna bağlı).
- **Name:** Yöneticinin adı, zorunlu bir metin alanı.
- **Surname:** Yöneticinin soyadı, zorunlu bir metin alanı.

2.4 Roller (Roles):

- **ID:** Birincil anahtar (PK), benzersiz rol kimliği.
- **Role:** Kullanıcının sahip olduğu yetki.

2.5 Rol İzinleri (RolePermissions):

- **ID:** Birincil anahtar (PK), benzersiz rol izni kimliği.
- **Role ID:** İzinlerin bağlı olduğu rol (FK, Roles tablosuna bağlı).
- **Table Name:** İzinlerin verildiği tablo adı.
- **CanRead, CanWrite, CanUpdate, CanDelete:** İlgili tablodaki yapılabilecek işlemleri bit türünde belirtir.

2.6 Departmanlar (Departments):

- **ID:** Birincil anahtar (PK), benzersiz departman kimliği.
- **Department Name:** Departman adı, örneğin "Matematik", "Fizik" gibi.

2.7 Dersler (Courses):

- **ID:** Birincil anahtar (PK), benzersiz ders kimliği.
- **Course Name:** Dersin adı, örneğin "Matematik", "Fizik" gibi.

2.8 Sınıflar (Classes):

- **ID:** Birincil anahtar (PK), benzersiz sınıf kimliği.
- **Class Name:** Sınıfın adı, örneğin "BD4", "BD7" gibi.

2.9 Günler (Days):

- **ID:** Birincil anahtar (PK), benzersiz gün kimliği.
- **Day Name:** Günün adı, örneğin "Pazartesi", "Salı" gibi.

2.10 Ders Programları (TimeTables):

- **ID:** Birincil anahtar (PK), benzersiz ders programı kimliği.
- **Course ID:** Ders programının bağlı olduğu ders (FK, Courses tablosuna bağlı).
- **Class ID:** Ders programının bağlı olduğu sınıf (FK, Classes tablosuna bağlı).
- **Day ID:** Ders programının bağlı olduğu gün (FK, Days tablosuna bağlı).
- **Start Time:** Dersin başlangıç saati.
- **End Time:** Dersin bitiş saati.

2.11 Yoklamalar (Attendances):

- **ID:** Birincil anahtar (PK), benzersiz yoklama kimliği.
- **Course ID:** Yoklamanın bağlı olduğu ders (FK, Courses tablosuna bağlı).
- **Date:** Yoklama yapılan tarih.
- **Status:** Devamsızlık durumu ("Present", "Absent", "Late", "Excused" gibi değerler alabilir).

2.12 Sınavlar (Exams):

- **ID:** Birincil anahtar (PK), benzersiz sınav kimliği.
- **Course ID:** Sınavın bağlı olduğu ders (FK, Courses tablosuna bağlı).
- **Date:** Sınavın yapıldığı tarih.
- **Exam Type:** Sınav türü, örneğin "Midterm", "Final", "Quiz".

2.13 Notlar (Grades):

- **ID:** Birincil anahtar (PK), benzersiz not kimliği.
- **Student ID:** Notun bağlı olduğu öğrenci (FK, Students tablosuna bağlı).
- **Exam ID:** Notun bağlı olduğu sınav (FK, Exams tablosuna bağlı).
- **Score:** Sınav puanı (numerik değer).

2.14 Kayıtlar (Logs):

- **ID:** Birincil anahtar (PK), benzersiz kayıt kimliği.
- **User Type:** Kullanıcının türü, örneğin "Öğrenci", "Öğretmen", "Yönetici".
- **User ID:** İşlemi gerçekleştiren kullanıcının ID'si.
- **Action:** Yapılan işlemin adı, örneğin "Ekleme", "Silme", "Güncelleme".
- **Description:** İşlemle ilgili açıklama.
- **Timestamp:** İşlemin yapıldığı tarih ve saat.

2.15 Öğrenci - Devamsızlık (StudentAttendance):

- **Student ID:** Birincil anahtar (PK), yoklamanın bağlı olduğu öğrenci.
- **Attendance ID:** Birincil anahtar (PK), öğrencinin bağlı olduğu yoklama.

2.16 Departman – Ders (DepartmentCourse):

- **Department ID:** Birincil anahtar (PK), dersin bağlı olduğu departman.
- **Course ID:** Birincil anahtar (PK), departmanın bağlı olduğu ders.

2.17 Öğretmen – Ders (TeacherCourse):

- **Teacher ID:** Birincil anahtar (PK), dersin bağlı olduğu öğretmen.
- **Course ID:** Birincil anahtar (PK), öğretmenin bağlı olduğu ders.

3. İlişikler ve Kısıtlamalar

3.1 Öğrenciler ve Departmanlar:

Her öğrenci yalnızca bir departmana atanır ve her departmanın birden fazla öğrencisi vardır. (N : 1)

3.2 Öğrenciler ve Öğretmenler:

Her öğrenci yalnızca bir danışmana atanır ve her danışmanın birden fazla öğrencisi vardır. (N : 1)

3.3 Öğrenciler ve Roller:

Her öğrenci yalnızca öğrenci rolüne atanır ve öğrenci rolünün birden fazla öğrencisi vardır. (N : 1)

3.4 Öğretmenler ve Departmanlar:

Her öğretmen yalnızca bir departmana atanır ve her departmanın birden fazla öğretmeni vardır. (N : 1)

3.5 Öğretmenler ve Roller:

Her öğretmen yalnızca öğretmen rolüne atanır ve öğretmen rolünün birden fazla öğretmeni vardır. (N : 1)

3.6 Yöneticiler ve Roller:

Her yönetici yalnızca yönetici rolüne atanır ve yönetici rolünün birden fazla yöneticisi vardır. (N : 1)

3.7 Rol İzinleri ve Roller:

Her rol izni yalnızca bir role atanır ve her rolün birden fazla rol izni vardır. (N : 1)

3.8 Ders Programları ve Dersler:

Her ders programı yalnızca bir derse atanır ve her dersin yalnızca bir ders programı vardır. (1 : 1)

3.9 Ders Programları ve Sınıflar:

Her ders programı yalnızca bir sınıfa atanır ve her sınıfın birden fazla ders programı vardır. (N : 1)

3.10 Ders Programları ve Günler:

Her ders programı yalnızca bir güne atanır ve her günün birden fazla ders programı vardır. (N : 1)

3.11 Yoklamalar ve Dersler:

Her yoklama yalnızca bir derse atanır ve her ders de yalnızca bir yoklamaya atanır. (1 : 1)

3.12 Sınavlar ve Dersler:

Her sınav yalnızca bir derse atanır ve her dersin birden fazla sınavı vardır. (N : 1)

3.13 Notlar ve Öğrenciler:

Her not yalnızca bir öğrenciye atanır ve her öğrencinin birden fazla notu vardır. (N : 1)

3.14 Notlar ve Sınavlar:

Her not yalnızca bir sınava atanır ve her sınavın birden fazla notu vardır. (N : 1)

3.15 Öğrenci – Devamsızlık:

Her öğrenci birden fazla yoklamaya atanır ve her yoklamanın birden fazla öğrencisi vardır. (N : M)

3.16 Departman – Ders:

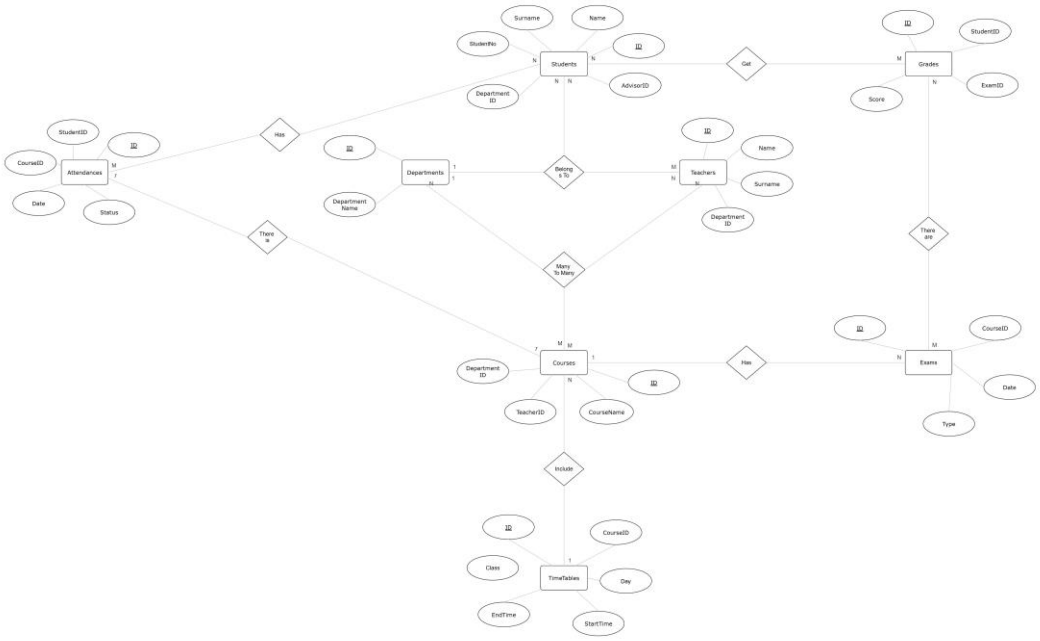
Her departman birden fazla derse atanır ve her dersin birden fazla departmanı vardır. (N : M)

3.17 Öğretmen – Ders:

Her öğretmen birden fazla derse atanır ve her dersin birden fazla öğretmeni vardır. (N : M)

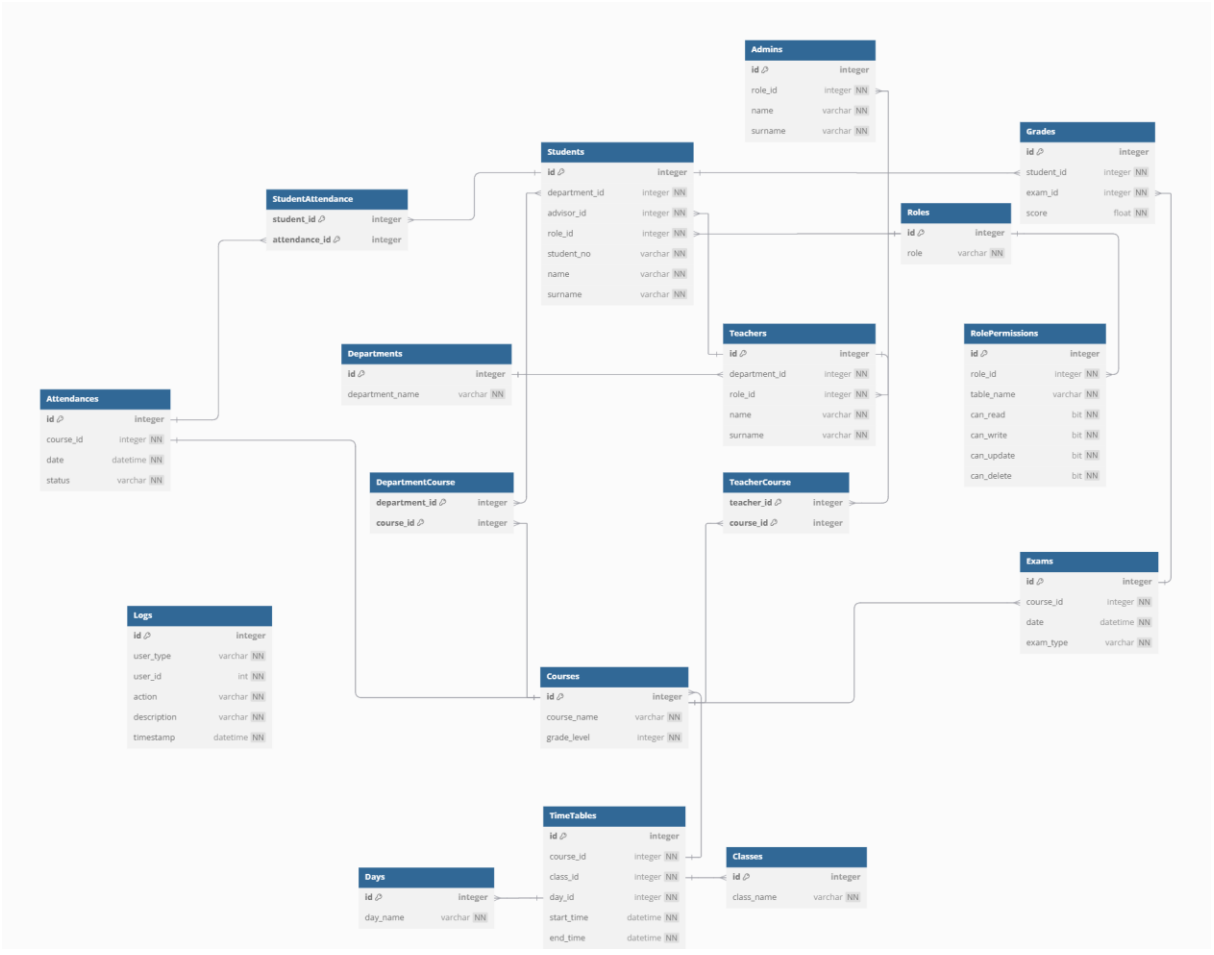
E – R Diyagramının Oluşturulması

Veri tabanı gereksinimleri göre belirlenen varlıklar, nitelikler ve ilişkilerle oluşturulan E – R diyagramı aşağıdaki gibi şekillendirilmiştir.



İlişkisel Veri Modelinin Oluşturulması

Veri tabanı gereksinimleri ve E-R diyagramına göre, veri türleri, NOT NULL ve UNIQUE kısıtlamalarına göre belirlenen tablo yapıları, sütunlar ve ilişkilerle oluşturulan veri tabanı tasarımının ilişkisel veri modeli aşağıdaki gibi şekillendirilmiştir.



Tabloların 3NF Olarak Normalleştirilmesi

1. **Öğrenciler (Students):**
   Öğrenciler tablosu, öğrenci kimliği, adı, soyadı, bağlı olduğu departman, danışman ve rol gibi temel bilgileri içermektedir. Bu tablo, her öğrenci için benzersiz bir ID (birincil anahtar) kullanarak, gereksiz veri tekrarını önler. Ayrıca, Department ID, Advisor ID ve Role ID gibi yabancı anahtarlar, başka tablolardaki ilişkilerle bağlıdır, bu da tablonun 3NF'ye uygunluğunu sağlar. Her öğrenciye ait bilgiler sadece öğrenciye özeldir ve başka bir tablodan türememektedir.

2. **Öğretmenler (Teachers):**
   Öğretmenler tablosunda, her öğretmen için benzersiz bir ID (birincil anahtar) kullanılarak, öğretmenin adı, soyadı, bağlı olduğu departman ve rol gibi bilgiler depolanmaktadır. Bu tablo, öğretmenin yalnızca bir departmana ve role atanması gerektiği için, 3NF gereksinimlerini karşılar. Öğretmenlerin bağlı olduğu departman ve rol bilgileri başka tablolarda tutulduğundan, burada veri tekrarını engeller.

3. **Yöneticiler (Admins):**
   Yöneticiler tablosu, her yönetici için benzersiz bir ID'yi (birincil anahtar) içerir ve yönetici bilgileri, yalnızca bu tablodaki satırlara özeldir. Role ID ise, yöneticiye atanan rolü belirtir ve rol bilgisi başka bir tabloda tutulur, bu da veri tekrarını engeller. Tablonun yapısı 3NF'ye uygundur.

4. **Roller (Roles):**
   Roller tablosu, her rol için benzersiz bir ID'yi (birincil anahtar) tutar ve yalnızca rol bilgilerini içerir. Öğrenciler, öğretmenler ve yöneticiler gibi farklı varlıklar bu tabloya yabancı anahtar (FK) ile bağlanır. Bu tablo, sadece tek bir tür bilgi depoladığı için 3NF gereksinimlerine uygundur.

5. **Rol İzinleri (RolePermissions):**
   Rol İzinleri tablosu, her rol için benzersiz bir ID'yi (birincil anahtar) içerir ve kullanıcı rollerine atanmış izinlerin yönetilmesini sağlar. Bu tablo, her rolün hangi işlemleri (okuma, yazma, güncelleme, silme) yapabileceğini belirtir. RoleID yabancı anahtar (FK) olarak Roles tablosuna bağlanır, bu da veri tekrarını engeller. Bu tablo, yalnızca rol izin bilgilerini içerdiğinden 3NF'ye uygundur.

6. **Departmanlar (Departments):**
   Departmanlar tablosu, her departman için benzersiz bir ID'yi (birincil anahtar) içerir ve sadece departman adı gibi temel bilgileri depolar. Departmanlar, öğrenci ve öğretmen gibi varlıklarla ilişkilendirildiği için, bu tablonun 3NF'ye uygunluğu sağlanır. Tek bir varlık (departman) ile ilgili olduğu için veri tekrarına neden olmaz.

7. **Dersler (Courses):**
   Dersler tablosu, her ders için benzersiz bir ID'yi (birincil anahtar) tutar ve ders adı gibi bilgileri içerir. Bu bilgiler yalnızca bu tabloda yer alır ve başka tablolardan türetilmez. Bu tablo, derslerle ilgili bilgileri tutarak, gereksiz veri tekrarını engeller ve 3NF'ye uygundur.

8. **Sınıflar (Classes):**
   Sınıflar tablosunda her sınıf için benzersiz bir ID (birincil anahtar) ve sınıf adı gibi bilgiler tutulur. Her sınıf yalnızca bir kez kaydedildiği için, veri tekrarını engeller ve bu tablo 3NF'ye uygundur.

9. **Günler (Days):**
   Günler tablosu, her gün için benzersiz bir ID'yi (birincil anahtar) içerir ve yalnızca gün adını tutar. Diğer tablolarda sadece gün bilgisi kullanılır, bu da gereksiz veri tekrarını önler ve 3NF'ye uygundur.

10. **Ders Programları (TimeTables):**
    Ders Programları tablosu, ders, sınıf ve gün bilgilerini içerir. Her ders programı benzersizdir ve bu bilgileri başka tablolardan alır. Bu yapılar, 3NF'ye uygundur, çünkü her bir sütun yalnızca bir tür bilgiye odaklanır ve tekrarlanan veri yoktur.

11. **Yoklamalar (Attendances):**
    Yoklamalar tablosu, her yoklamayı benzersiz bir şekilde tanımlar ve her yoklama sadece bir dersle ilişkilidir. Bu tablo, ders bilgilerini bir yabancı anahtar aracılığıyla alır ve veri tekrarını engeller, böylece 3NF'ye uygun olur.

12. **Sınavlar (Exams):**
    Sınavlar tablosu, her sınav için benzersiz bir ID'yi (birincil anahtar) tutar ve sınav türü gibi bilgileri içerir. Sınavlar, yalnızca bir derse atanır ve her sınavda birden fazla not olabilir. Bu da 3NF'ye uygunluğu sağlar.

13. **Notlar (Grades):**
    Notlar tablosu, her öğrenciye ait sınav puanlarını depolar ve yalnızca öğrenci ve sınav bilgileri ile ilgilidir. Öğrencinin sınavla ilişkisi ve puan bilgisi başka tablolardan alınır, böylece veri tekrarını engeller ve 3NF'ye uygundur.

14. **Kayıtlar (Logs):**
    Kayıtlar tablosu, kullanıcıların yaptığı işlemleri takip etmek amacıyla oluşturulmuştur. Bu tablo, her işlem için benzersiz bir ID (birincil anahtar) kullanır ve işlemle ilgili bilgileri içerir. UserID yabancı anahtar (FK) olarak, kullanıcıyı belirten başka bir tabloya bağlanabilir. Ayrıca, her kaydın bir açıklaması ve zaman damgası da bulunur. Bu yapı, işlemle ilgili yalnızca gerekli bilgileri tutar ve 3NF'ye uygundur.

15. **Öğrenci - Devamsızlık (StudentAttendance):**
    Bu ilişki tablosu, öğrenci ve yoklama bilgilerini tutar. Bir öğrenci birden fazla yoklamaya atanabilir, ancak her yoklama birden fazla öğrenciye atanabilir. Bu tablo, ilişkili verileri birbirinden ayırır ve 3NF'ye uygundur.

16. **Departman - Ders (DepartmentCourse):**
    Departman ve ders arasındaki ilişkiyi belirler. Her departman birden fazla derse sahip olabilir, ancak her ders birden fazla departmanla ilişkilendirilebilir. Bu tablo, ilişkili verileri ayırarak, 3NF'ye uygundur.

17. **Öğretmen - Ders (TeacherCourse):**
    Öğretmen ve ders arasındaki ilişkiyi tanımlar. Her öğretmen birden fazla derse atanabilir ve her ders birden fazla öğretmenle ilişkilendirilebilir. Bu tablo da 3NF'ye uygundur çünkü ilişkili veriler tek bir yerde tutulur.

SQL Server'da Tablo, İlişki ve Kısıtlamaların Eklenmesi ve Veritabanı Şemasının Oluşturulması

Bu bölümde, veri tabanı tasarımının SQL Server kullanılarak tablolarda gerekli kısıtlamaların ve ilişkilerin nasıl oluşturulacağına dair adımlar anlatılmaktadır. Öncelikle, her bir tablonun oluşturulması için `CREATE TABLE` komutları kullanılarak veri tabanı şeması tasarlanmıştır. Ardından, tablolar arasındaki ilişkiler (FOREIGN KEY) belirlenmiş ve uygun kısıtlamalar (`PRIMARY KEY`, `UNIQUE`, `NOT NULL`) eklenmiştir.

Aşağıda, tablolarda yer alan sütunlar, veri türleri ve her bir tablonun ilişki yapılarını içeren SQL kodları yer almaktadır. Bu kodlar, veri tabanının doğru şekilde yapılandırılmasını sağlayarak verilerin tutarlılığını ve entegrasyonunu garanti altına alır.

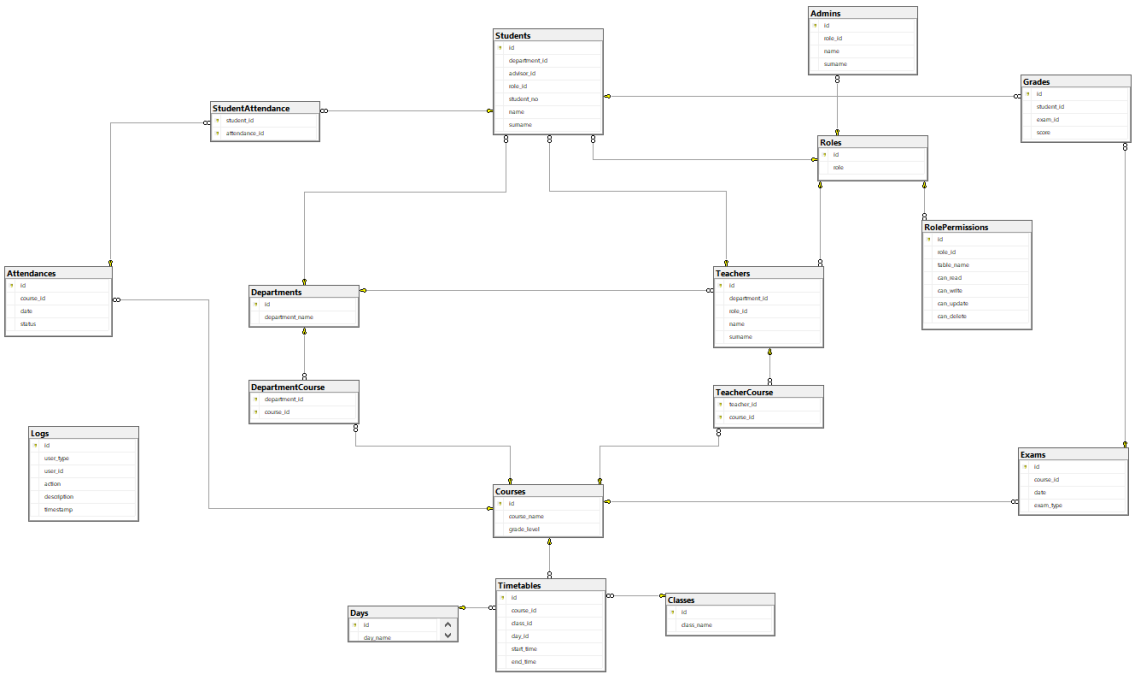
```sql
-- Tabloların Oluşturulması ve İlişkilerin Tanımlanması
-- 1. Departments Tablosu
CREATE TABLE Departments (
    ID INT PRIMARY KEY IDENTITY(1,1),
    DepartmentName NVARCHAR(255) NOT NULL
);

-- 2. Roles Tablosu
CREATE TABLE Roles (
    ID INT PRIMARY KEY IDENTITY(1,1),
    Role NVARCHAR(255) NOT NULL
);

-- 3. Teachers Tablosu
CREATE TABLE Teachers (
    ID INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(255) NOT NULL,
    Surname NVARCHAR(255) NOT NULL,
    DepartmentID INT NOT NULL,
    RoleID INT NOT NULL,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(ID),
    FOREIGN KEY (RoleID) REFERENCES Roles(ID)
);
```

```sql
-- 4. Students Tablosu
CREATE TABLE Students (
    ID INT PRIMARY KEY IDENTITY(1,1),
    StudentNo NVARCHAR(50) NOT NULL UNIQUE,
    Name NVARCHAR(255) NOT NULL,
    Surname NVARCHAR(255) NOT NULL,
    DepartmentID INT NOT NULL,
    AdvisorID INT,
    RoleID INT NOT NULL,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(ID),
    FOREIGN KEY (AdvisorID) REFERENCES Teachers(ID),
    FOREIGN KEY (RoleID) REFERENCES Roles(ID)
);

-- 5. Admins Tablosu
CREATE TABLE Admins (
    ID INT PRIMARY KEY IDENTITY(1,1),
    RoleID INT NOT NULL,
    name NVARCHAR(255) NOT NULL,
    surname NVARCHAR(255) NOT NULL,
    FOREIGN KEY (RoleID) REFERENCES Roles(ID)
);

-- 6. RolePermissions Tablosu
CREATE TABLE RolePermissions (
    ID INT PRIMARY KEY IDENTITY(1,1),
    RoleID INT NOT NULL,
    TableName NVARCHAR(255) NOT NULL,
    CanRead BIT DEFAULT 0,
    CanWrite BIT DEFAULT 0,
    CanUpdate BIT DEFAULT 0,
    CanDelete BIT DEFAULT 0,
    FOREIGN KEY (RoleID) REFERENCES Roles(ID)
);

-- 7. Courses Tablosu
CREATE TABLE Courses (
    ID INT PRIMARY KEY IDENTITY(1,1),
    CourseName NVARCHAR(255) NOT NULL,
       GradeLevel INT
);

-- 8. Classes Tablosu
CREATE TABLE Classes (
    ID INT PRIMARY KEY IDENTITY(1,1),
    ClassName NVARCHAR(255)
);

-- 9. Days Tablosu
CREATE TABLE Days (
    ID INT PRIMARY KEY IDENTITY(1,1),
    DayNames NVARCHAR(255)
);
```

```sql
-- 10. Timetables Tablosu
CREATE TABLE Timetables (
    ID INT PRIMARY KEY IDENTITY(1,1),
    CourseID INT NOT NULL,
    ClassID INT NOT NULL,
    DayID INT NOT NULL,
    StartTime DATETIME NOT NULL,
    EndTime DATETIME NOT NULL,
    FOREIGN KEY (CourseID) REFERENCES Courses(ID),
    FOREIGN KEY (ClassID) REFERENCES Classes(ID),
    FOREIGN KEY (DayID) REFERENCES Days(ID)
);

-- 11. Attendances Tablosu
CREATE TABLE Attendances (
    ID INT PRIMARY KEY IDENTITY(1,1),
    Date DATE NOT NULL,
    Status NVARCHAR(50) NOT NULL
);

-- 12. Exams Tablosu
CREATE TABLE Exams (
    ID INT PRIMARY KEY IDENTITY(1,1),
       CourseID INT NOT NULL,
    Date DATE NOT NULL,
    ExamType NVARCHAR(50) NOT NULL,
       FOREIGN KEY (CourseID) REFERENCES Courses(ID)
);

-- 13. Grades Tablosu
CREATE TABLE Grades (
    ID INT PRIMARY KEY IDENTITY(1,1),
    Score FLOAT NOT NULL,
       StudentID INT NOT NULL,
       ExamID INT NOT NULL,
       FOREIGN KEY (StudentID) REFERENCES Students(ID),
       FOREIGN KEY (ExamID) REFERENCES Exams(ID)
);

-- 14. Logs Tablosu
CREATE TABLE Logs (
    ID INT PRIMARY KEY IDENTITY(1,1),
    UserType NVARCHAR(50), -- Kullanıcı türü (Öğrenci, Öğretmen, Yönetici)
    UserID INT, -- İlgili kullanıcı ID'si
    Action NVARCHAR(50) NOT NULL,
    Description NVARCHAR(255),
    Timestamp DATETIME DEFAULT GETDATE()
);

-- 15. İlişki Tabloları
CREATE TABLE StudentAttendances (
    StudentID INT NOT NULL,
    AttendanceID INT NOT NULL,
    PRIMARY KEY (StudentID, AttendanceID),
    FOREIGN KEY (StudentID) REFERENCES Students(ID),
    FOREIGN KEY (AttendanceID) REFERENCES Attendances(ID)
);
```

```sql
CREATE TABLE CourseDepartments (
    CourseID INT NOT NULL,
    DepartmentID INT NOT NULL,
    PRIMARY KEY (CourseID, DepartmentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(ID),
    FOREIGN KEY (DepartmentID) REFERENCES Departments(ID)
);

CREATE TABLE CourseTeachers (
    CourseID INT NOT NULL,
    TeacherID INT NOT NULL,
    PRIMARY KEY (CourseID, TeacherID),
    FOREIGN KEY (CourseID) REFERENCES Courses(ID),
    FOREIGN KEY (TeacherID) REFERENCES Teachers(ID)
);
```

Aşağıda yer alan SQL Server diyagramı, yazılan SQL kodlarıyla oluşturulan tablo ve ilişkilerin görsel bir temsilidir. Diyagram, tablolar arasındaki dış anahtar ilişkilerini ve veri modelinin 3NF'ye uygunluğunu doğrulamaktadır.



SQL Server'da Tablolara Veri Eklenmesi

Aşağıda, SQL kodlarıyla veritabanına veri eklemek için kullanılan INSERT INTO komutları yer almaktadır. Bu komutlar, önceden oluşturulmuş tablolara, her bir tablonun belirlediği sütunlara uygun verileri eklemek için kullanılacaktır. Verilerin fazlalığı nedeniyle her tablodan kısa veriler rapora eklenmiştir. Bu işlemler, veritabanındaki varlıkların ve ilişkilerin doğru şekilde oluşturulması ve veri modeline uygunluğun sağlanması açısından önemlidir.

```sql
-- 1. Departments Tablosu İçin Veri Ekleme
INSERT INTO Departments (department_name) VALUES
(N'Bilgisayar Mühendisliği'),
(N'Elektrik Elektronik Mühendisliği'),
(N'Makine Mühendisliği'),
(N'İnşaat Mühendisliği'),
(N'Endüstri Mühendisliği'),
(N'Kimya Mühendisliği'),
(N'Mimarlık'),
(N'Tıp'),
(N'Hukuk'),
(N'İktisat');


-- 2. Roles Tablosu İçin Veri Ekleme
INSERT INTO Roles (role) VALUES
(N'Admin'),
(N'Teacher'),
(N'Student');



-- 3. Teachers Tablosu İçin Veri Ekleme
INSERT INTO Teachers (department_id, role_id, name, surname) VALUES
(1, 2, N'Ertan', N'Bütün'),
(1, 2, N'Burhan', N'Ergen'),
(1, 2, N'Ahmet', N'Çınar'),
(1, 2, N'Güngör', N'Yıldırım'),
(1, 2, N'Murat', N'Aydın'),
(1, 2, N'Ayşe', N'Çelik'),
(1, 2, N'Ahmet', N'Şahin'),
(2, 2, N'Fatma', N'Demir'),
(2, 2, N'Murat', N'Arslan'),
(2, 2, N'Mehmet', N'Kaya');



-- 4. Students Tablosu İçin Veri Ekleme
INSERT INTO Students (department_id, advisor_id, role_id, student_no, name, surname)
VALUES
(1, 1, 3, N'220260001', N'Ali', N'Yıldız'),
(2, 8, 3, N'220260002', N'Aslı', N'Demir'),
(3, 15, 3, N'220260003', N'Emre', N'Kurt'),
(4, 22, 3, N'220260004', N'Canan', N'Soylu'),
(5, 29, 3, N'220260005', N'Yusuf', N'Güler'),
(6, 36, 3, N'220260006', N'Mehmet Miraç', N'Özmen'),
(7, 43, 3, N'220260007', N'Efe', N'Koç'),
(8, 50, 3, N'220260008', N'Berna', N'Tekin'),
(9, 57, 3, N'220260009', N'Ozan', N'Taş'),
(10, 64, 3, N'220260010', N'Işıl', N'Özkan');



-- 5. Admins Tablosu İçin Veri Ekleme
INSERT INTO Admins (role_id, name, surname) VALUES
(1, N'Miraç', N'Özmen'),
(1, N'Alperen', N'Aktaş'),
(1, N'Emre', N'Gümüş');
```

```sql
-- 6. Courses Tablosu İçin Veri Ekleme
INSERT INTO Courses (course_name, grade_level) VALUES
(N'Veri Yapıları ve Algoritmalar', 1),
(N'Yapay Zeka ve Makine Öğrenimi', 1),
(N'Web Programlama', 2),
(N'Mobil Programlama', 2),
(N'Veri Tabanı Yönetimi', 3),
(N'Mikroişlemciler', 3),
(N'Gömülü Sistemler Tasarımı', 4),
(N'Bilgisayar Grafiklerine Giriş', 4);


-- 7. Classes Tablosu İçin Veri Ekleme
INSERT INTO Classes (class_name) VALUES
(N'BM 1'),
(N'BM 2'),
(N'BM 3'),
(N'BM 4'),
(N'BM 5');



-- 8. Days Tablosu İçin Veri Ekleme
INSERT INTO Days (day_name) VALUES
('Pazartesi'),
('Salı'),
('Çarşamba'),
('Perşembe'),
('Cuma');



-- 9. Timetables Tablosu İçin Veri Ekleme
INSERT INTO Timetables (course_id, class_id, day_id, start_time, end_time) VALUES
(1, 1, 1, '08:00', '10:00'),
(2, 1, 1, '10:00', '12:00'),
(3, 2, 1, '10:00', '12:00'),
(4, 1, 1, '12:00', '14:00'),
(1, 2, 2, '08:00', '10:00'),
(5, 3, 2, '08:00', '10:00'),
(3, 2, 2, '14:00', '16:00'),
(6, 5, 2, '14:00', '16:00'),
(1, 2, 3, '08:00', '10:00'),
(7, 3, 3, '10:00', '12:00');



-- 10. Attendances Tablosu İçin Veri Ekleme
INSERT INTO Attendances (course_id, date, status) VALUES
(1, '2024-10-01', N'Katıldı'),
(2, '2024-10-02', N'Katıldı'),
(3, '2024-10-03', N'Katılmadı'),
(4, '2024-10-04', N'Katıldı'),
(5, '2024-10-06', N'Katıldı'),
(6, '2024-10-07', N'Katılmadı'),
(7, '2024-10-08', N'Katıldı'),
(8, '2024-10-09', N'Katıldı'),
(9, '2024-10-10', N'Katılmadı'),
(10, '2024-10-12', N'Katıldı');
```

```sql
-- 11. Exams Tablosu İçin Veri Ekleme
INSERT INTO Exams (course_id, date, exam_type) VALUES
(1,'2024-01-01', N'Ara Sınav'),
(2,'2024-01-05', N'Final'),
(9,'2024-01-10', N'Quiz'),
(10,'2024-01-15', N'Vize'),
(17,'2024-01-20', N'Proje'),
(18,'2024-01-25', N'Ara Sınav'),
(25,'2024-01-30', N'Final'),
(26,'2024-02-01', N'Quiz'),
(35,'2024-02-05', N'Vize'),
(36,'2024-02-10', N'Proje'),
(45,'2024-02-15', N'Ara Sınav');



-- 12. Kısıtlar için bilgilendirme verileri
-- Yöneticiler
INSERT INTO RolePermissions (RoleID, TableName, CanRead, CanWrite, CanUpdate,
CanDelete)
VALUES
(1, 'Students', 1, 1, 1, 1),
(1, 'Teachers', 1, 1, 1, 1),
(1, 'Courses', 1, 1, 1, 1),
(1, 'Timetables', 1, 1, 1, 1),
(1, 'Grades', 1, 1, 1, 1),
(1, 'Attendances', 1, 1, 1, 1),
(1, 'Logs', 1, 1, 1, 1);

-- Öğretmenler
INSERT INTO RolePermissions (RoleID, TableName, CanRead, CanWrite, CanUpdate,
CanDelete)
VALUES
(2, 'Courses', 1, 1, 1, 0),
(2, 'Grades', 1, 1, 1, 0),
(2, 'Attendances', 1, 1, 1, 0);

-- Öğrenciler
INSERT INTO RolePermissions (RoleID, TableName, CanRead, CanWrite, CanUpdate,
CanDelete)
VALUES
(3, 'Students', 1, 0, 0, 0),
(3, 'Timetables', 1, 0, 0, 0),
(3, 'Grades', 1, 0, 0, 0),
(3, 'Attendances', 1, 0, 0, 0);


-- 13. Grades Tablosu İçin Veri Ekleme
INSERT INTO Grades (student_id, exam_id, score) VALUES
(1, 1, 45), (2, 2, 67), (3, 3, 82), (4, 4, 59), (5, 5, 36),
(6, 6, 78), (7, 7, 92), (8, 8, 88), (9, 9, 49), (10, 10, 73);



-- 14. İlişki Tabloları İçin Veri Ekleme

-- StudentAttendances Tablosu İçin Veri Ekleme
INSERT INTO StudentAttendance (student_id, attendance_id) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
(6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
```

```sql
-- DepartmentCourse Tablosu İçin Veri Ekleme
INSERT INTO DepartmentCourse (department_id, course_id) VALUES
(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8),
(2, 9), (2, 10), (2, 11), (2, 12), (2, 13), (2, 14), (2, 15), (2, 16);


-- TeacherCourse Tablosu İçin Veri Ekleme
INSERT INTO TeacherCourse (teacher_id, course_id) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7),
(9, 9), (10, 10), (11, 11), (12, 12), (13, 13), (14, 14), (15, 15);
```

SQL Server'da Kısıtlamalar ve Yetkilendirmelerin Eklenmesi

1. **Öğrenciler:**
   - Kendi tablolarını görüntüleyebilir.
   - Devamsızlık kayıtlarını ve sınav sonuçlarını görebilir.
   - Ders programına ve sınav takvimine erişebilir.
   - Yalnızca okuma yetkisine sahiptir, veri girişi yapamaz.

2. **Öğretmenler:**
   - Kendi verdikleri derslerin öğrenci listelerine, devamsızlık kayıtlarına ve sınav sonuçlarına erişebilirler.
   - Dersleri için sınav oluşturabilir ve sınav sonuçlarını güncelleyebilirler.
   - Öğrencilerin ders içindeki performans durumunu görebilirler.
   - Hem okuma hem yazma yetkisine sahiptir.

3. **Yöneticiler:**
   - Tüm öğrenci ve öğretmen kayıtlarını yönetme yetkisine sahiptir.
   - Yeni dersler oluşturabilir, ders programlarını ve sınavları düzenleyebilir.
   - Öğrencilerin ve öğretmenlerin bilgilerini güncelleyebilir veya sistemden çıkarabilirler.
   - Veri tabanındaki tüm verilere tam erişim (okuma, yazma, güncelleme ve silme) yetkisine sahiptir.

Belirtilen kısıtların eklenme komutları aşağıda belirtilmiştir.

```sql
-- 1. Rolleri oluştur.
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'StudentRole')
    CREATE ROLE StudentRole;

IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'TeacherRole')
    CREATE ROLE TeacherRole;

IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'AdminRole')
    CREATE ROLE AdminRole;

-- 2. Kullanıcılar için login'ler oluştur (SQL Server oturumları)
CREATE LOGIN student1 WITH PASSWORD = 'student';
CREATE LOGIN teacher1 WITH PASSWORD = 'teacher';
CREATE LOGIN admin1 WITH PASSWORD = 'admin';
```

```sql
USE OBS;

-- 4. Veritabanında kullanıcıları oluştur
CREATE USER student1 FOR LOGIN student1;
CREATE USER teacher1 FOR LOGIN teacher1;
CREATE USER admin1 FOR LOGIN admin1;

-- 5. Kullanıcıları rollere ekle
ALTER ROLE StudentRole ADD MEMBER student1;
ALTER ROLE TeacherRole ADD MEMBER teacher1;
ALTER ROLE AdminRole ADD MEMBER admin1;

-- 6. Öğrenci Rolü: Öğrencilerin erişebileceği izinler
GRANT SELECT ON Students TO StudentRole;
GRANT SELECT ON Timetables TO StudentRole;
GRANT SELECT ON Exams TO StudentRole;
GRANT SELECT ON Attendances TO StudentRole;
GRANT SELECT ON Grades TO StudentRole;
GRANT SELECT ON RolePermissions TO StudentRole;


-- 7. Öğretmen Rolü: Öğretmenlerin erişebileceği izinler
GRANT SELECT, INSERT, UPDATE ON Courses TO TeacherRole;
GRANT SELECT, INSERT, UPDATE ON Attendances TO TeacherRole;
GRANT SELECT, INSERT, UPDATE ON Grades TO TeacherRole;
GRANT SELECT ON RolePermissions TO TeacherRole;
GRANT SELECT ON Students TO TeacherRole;

-- 8. Yönetici Rolü: Yöneticilerin tüm tablolarda tam yetkiye sahip olması
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Departments TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Roles TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Teachers TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Students TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Admins TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON RolePermissions TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Courses TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Classes TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Days TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Timetables TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Attendances TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Exams TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Grades TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON Logs TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON StudentAttendances TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON CourseDepartments TO AdminRole;
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER ON CourseTeachers TO AdminRole;
```

SQL Server’da Gereksinimlerin Testi için Örnek Komutlar

1: Tüm öğretmenlerin isimlerini ve ait oldukları bölümün isimlerini listele.

```sql
SELECT t.Name, t.Surname, d.DepartmentName
FROM Teachers t
JOIN Departments d ON t.DepartmentID = d.ID;
```

2: **Tüm öğrencilerin danışman öğretmenlerinin isimleriyle birlikte listelenmesi.**

```
SELECT s.Name AS StudentName, s.Surname AS StudentSurname, t.Name AS AdvisorName,
t.Surname AS AdvisorSurname
FROM Students s
LEFT JOIN Teachers t ON s.AdvisorID = t.ID;
```

3: Her bölümdeki toplam öğrenci sayısını getir.

```
SELECT d.DepartmentName, COUNT(s.ID) AS TotalStudents
FROM Departments d
LEFT JOIN Students s ON d.ID = s.DepartmentID
GROUP BY d.DepartmentName;
```

4: Tüm dersleri ve bu derslerin ait olduğu bölümleri listele.

```
SELECT c.CourseName, d.DepartmentName
FROM Courses c
JOIN CourseDepartments cd ON c.ID = cd.CourseID
JOIN Departments d ON cd.DepartmentID = d.ID;
```

5: **Tüm öğretmenleri ve verdikleri dersleri listele.**

```
SELECT t.Name AS TeacherName, t.Surname AS TeacherSurname, c.CourseName
FROM Teachers t
JOIN CourseTeachers ct ON t.ID = ct.TeacherID
JOIN Courses c ON ct.CourseID = c.ID;
```

6: **Belirli bir öğrencinin notlarını getir.**

```
SELECT g.Score, e.ExamType, e.Date
FROM Grades g
JOIN Exams e ON g.ExamID = e.ID
WHERE g.StudentID = 1;
```

7: Belirli bir tarihte tüm öğrencilerin devam durumlarını getir.

```
SELECT s.Name AS StudentName, s.Surname AS StudentSurname, a.Status
FROM Students s
JOIN StudentAttendances sa ON s.ID = sa.StudentID
JOIN Attendances a ON sa.AttendanceID = a.ID
WHERE a.Date = '2024-10-30';
```

8: **Tüm dersleri ve bu derslerin sınav bilgilerini listele.**

```
SELECT c.CourseName, e.ExamType, e.Date
FROM Courses c
JOIN Exams e ON c.ID = e.CourseID;
```

9: **Notu 50'nin altında olan tüm öğrencileri getir.**

```
SELECT s.Name AS StudentName, s.Surname AS StudentSurname, g.Score, e.ExamType
FROM Grades g
JOIN Students s ON g.StudentID = s.ID
JOIN Exams e ON g.ExamID = e.ID
```

```
WHERE g.Score < 50;
```

10: Belirli bir Departmanın tüm ders programını getir.

```
SELECT tt.StartTime, tt.EndTime, c.CourseName, cl.ClassName, d.DayNames
FROM Timetables tt
JOIN Days d ON tt.DayID = d.ID
JOIN Courses c ON tt.CourseID = c.ID
JOIN Classes cl ON tt.ClassID = cl.ID
JOIN CourseDepartments cd ON c.ID = cd.CourseID
WHERE cd.DepartmentID = 1
ORDER BY tt.StartTime;
```

SQL Server'da Saklı Yordam Prosedürünün Eklenmesi

Projemizdeki kritik bir iş operasyonu olan öğrenci ekleme işlemini saklı yordam prosedürü ile ele aldık. Bu prosedür, öğrenci kaydı işlemlerini gerçekleştiren bir saklı yordamdır. Öğrenci kaydını yapmadan önce aşağıdaki kontrolleri gerçekleştirir:

Verilen `DepartmentID` geçerli bir departmanla eşleşiyorsa işlem devam eder. Eğer geçerli bir departman yoksa, kayıt işlemi yapılmaz.

Verilen `AdvisorID` geçerli bir öğretmen ID'siyle eşleşiyorsa işlem devam eder. Eğer geçerli bir danışman yoksa, kayıt işlemi yapılmaz.

Verilen öğrenci numarasının (`StudentNo`) veritabanında daha önce kayıtlı olup olmadığı kontrol edilir. Eğer öğrenci numarası zaten kayıtlıysa, yeni kayıt yapılmaz.

Bu kontrollerin ardından, geçerli bir işlem olduğu takdirde öğrenci kaydı yapılır ve `RoleID` 3 olarak atanır. (projedeki kısıtlara göre öğrencilerin hepsinin roleid'si 3'tür.) Bu prosedür ile öğrenci kaydının güvenli bir şekilde gerçekleştirilebilmesi için gerekli tüm doğrulama işlemleri yapılmaktadır.

Aşağıda, SQL kodlarıyla veritabanına bu prosedürü eklemek için kullanılan komutlar ve test etmek için gerekli olan sorgu örneği yer almaktadır.

Prosedür Kodu :

```
CREATE PROCEDURE RegisterStudent
    @StudentNo NVARCHAR(50),
    @Name NVARCHAR(255),
    @Surname NVARCHAR(255),
    @DepartmentID INT,
    @AdvisorID INT
AS
BEGIN
    -- Geçerli bir department ID'sinin olup olmadığını kontrol et
    IF NOT EXISTS (
        SELECT 1
        FROM Departments
        WHERE ID = @DepartmentID
    )
    BEGIN
```

```
        PRINT 'Geçersiz departman ID. Öğrenci kaydı gerçekleştirilemiyor.';
        RETURN;
    END

    -- AdvisorID'nin geçerli bir öğretmen ID'si olup olmadığını kontrol et
    IF NOT EXISTS (
        SELECT 1
        FROM Teachers
        WHERE ID = @AdvisorID
    )
    BEGIN
        PRINT 'Geçersiz danışman ID. Öğrenci kaydı gerçekleştirilemiyor.';
        RETURN;
    END

    -- Öğrencinin benzersiz olup olmadığını kontrol et
    IF EXISTS (
        SELECT 1
        FROM Students
        WHERE StudentNo = @StudentNo
    )
    BEGIN
        PRINT 'Bu öğrenci numarası zaten kayıtlı.';
        RETURN;
    END

    -- Yeni öğrenci kaydı yap, RoleID her zaman 3 olacak
    INSERT INTO Students (StudentNo, Name, Surname, DepartmentID, AdvisorID, RoleID)
    VALUES (@StudentNo, @Name, @Surname, @DepartmentID, @AdvisorID, 3);

    PRINT 'Öğrenci başarıyla kaydedildi.';
END;
```

Test Sorgusu:

```
EXEC RegisterStudent
    @StudentNo = '12345',
    @Name = 'Ali',
    @Surname = 'Veli',
    @DepartmentID = 1, -- Geçerli bir departman ID'si

    @AdvisorID = 2; -- Geçerli bir öğretmen ID'si
```

SQL Server'da Tetikleyici Eklenmesi

Projemize eklediğimiz bu tetikleyici, `Students` tablosundan bir öğrenci silindiğinde otomatik olarak devreye girer. Silinen öğrenciyle ilgili bilgileri (ID, ad, soyad) ve işlem türünü (`DELETE`) `Logs` tablosuna kaydeder.

`AFTER DELETE` tetikleyicisi, `Students` tablosunda bir `DELETE` işlemi gerçekleştiğinde çalışır. Bu tetikleyici, silinen öğrenciyi `DELETED` sanal tablosundan alır ve `Logs` tablosuna kaydeder. `Logs` tablosuna, öğrencinin `ID`'si, işlem türü (`DELETE`) ve açıklama (öğrencinin adı ve soyadı ile birlikte "Öğrenci silindi" ifadesi) eklenir. Böylece sistemdeki her silme işlemi izlenebilir olur. Aşağıda kullanımıyla ve test sorgusuyla ilgili komutlar bulunmaktadır.

Tetikleyici Kodu :

```
CREATE TRIGGER trg_Student_Delete
ON Students
AFTER DELETE
AS
BEGIN
    -- Silinen öğrenciyi log tablosuna kaydet
    INSERT INTO Logs (UserType, UserID, Action, Description, Timestamp)
    SELECT
        'Student' AS UserType, -- Kullanıcı türü
        ID AS UserID,          -- Silinen öğrenci ID'si
        'DELETE' AS Action,    -- İşlem türü
        CONCAT('Öğrenci silindi: ', Name, ' ', Surname) AS Description, -- Açıklama
        GETDATE() AS Timestamp -- İşlem zamanı
    FROM
        DELETED;
END;
```

Test Sorgusu:

```
DELETE FROM Students

WHERE StudentNo = '12345';
```

Testten sonra Logs Tablosu görüntüsü:

Results | Messages

	ID	StudentID	Action	Description	Timestamp
1	1	401	DELETE	Öğrenci silindi: Ali Veli	2025-01-04 14:55:48.640

SQL Server'da Transaction Yönetimi Eklenmesi

Projemizdeki, birden fazla ilgili değişikliğin atomik olarak uygulanması gereken örnek bir senaryo olan eklenen yeni dersin not girişi işleminde;
Eğer girilen `DepartmentID` veya `TeacherID` veri tabanında geçerli değilse, işlem geri alınır (`ROLLBACK TRANSACTION`) ve uygun hata mesajı atanır.

Eğer `Score(not)` 0 ile 100 arasında değilse, işlem geri alınır ve bir hata mesajı döner.

Eğer işlemde bir hata oluşursa, `ROLLBACK TRANSACTION` ile tüm işlemler geri alınır, böylece veri tutarsızlıkları önlenir. Aşağıda örnek senaryo verilmiştir:

```
BEGIN TRANSACTION;

BEGIN TRY
    -- 1. Yeni bir ders ekle
    INSERT INTO Courses (CourseName, GradeLevel)
    VALUES ('Matematik 2', 3);

    -- Eklenen dersin ID'sini al
    DECLARE @NewCourseID INT;
    SET @NewCourseID = SCOPE_IDENTITY();
```

```
    -- 2. Bu dersin bir bölümle ilişkisini kur
    INSERT INTO CourseDepartments (CourseID, DepartmentID)
    VALUES (@NewCourseID, 1);

    -- 3. Bu dersin bir öğretmenle ilişkisini kur
    INSERT INTO CourseTeachers (CourseID, TeacherID)
    VALUES (@NewCourseID, 1);

    -- 4. Derse ait bir sınav ekle
    INSERT INTO Exams (CourseID, Date, ExamType)
    VALUES (@NewCourseID, '2025-01-15', 'Midterm');

    -- Eklenen sınavın ID'sini al
    DECLARE @NewExamID INT;
    SET @NewExamID = SCOPE_IDENTITY();

    -- 5. Öğrenci notu eklemeden önce notun geçerli olup olmadığını kontrol et
    DECLARE @Score FLOAT;
    SET @Score = 50;

    IF @Score < 0 OR @Score > 100
    BEGIN
        -- Hata mesajını konsola yazdır
        RAISERROR('Sınav notu 0 ile 100 arasında olmalıdır.', 16, 1);

        -- Hata fırlat
        THROW 50000, 'Sınav notu 0 ile 100 arasında olmalıdır.', 1;
    END

    -- Geçerli bir notsa, sınav notunu ekle
    INSERT INTO Grades (Score, StudentID, ExamID)
    VALUES (@Score, 1, @NewExamID); -- Örnek olarak StudentID = 1

    -- İşlemler başarılıysa transaction'ı onayla
    COMMIT TRANSACTION;

    PRINT 'İşlem başarıyla tamamlandı.';
END TRY

BEGIN CATCH
    -- Hata durumunda transaction'ı geri al
    ROLLBACK TRANSACTION;

    -- Hata detaylarını yakala ve mesaj göster
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

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

    -- Hata mesajını kullanıcıya göster
    PRINT 'Tüm işlemler geri alındı.(ROLLBACK)';
    RAISERROR ('Hata: %s', @ErrorSeverity, @ErrorState, @ErrorMessage);
END CATCH;
```

Açıklama : Yukarıdaki kodda hatalı TeacherID girişi olduğunda DepartmantID ve Score verileri doğru girilmesine rağmen ve not 0-100 aralığında yani uygun girilmesine rağmen veri tabanı, atomic uygulanması gereken bu işlemi rollback yaparak gerçekleştirmeyecektir. Başka bir senaryo olan not bilgisinin 0-100 aralığında olmadığı veri girişinde ise tüm diğer id verileri uygun olsa da atomic uygulanması gereken bu işlem rollback yapacaktır. Bu senaryolar diğer parametrelerin uyumsuzluğuyla da denenirse yine rollback mekanizması görüntülenebilecektir. Tüm verilerin doğru ve tablodaki verilerle uyumlu girildiği senaryoda ise tüm işlemler gerçekleştirilecek ve tüm tablolara tüm veriler eklenecektir. Aşağıda bu senaryoların çıktıları verilmiştir.