Veri Tabanı Yönetimi ve Modellemesi

HAFTA 10

Dr. Fatmana Şentürk

Haftalık Ders Akışı

1. Veritabanı Kavramlarına Giriş
2. Veri Tabanı Türleri, İlişkisel Veri Tabanı Tasarımı
3. ER Diyagramları ve Normalizasyon
4. SQL Server Arayüzü, Veri Tabanı Nesneleri
5. T-SQL ve SQL Sorguları
6. İndeks ve View
7. Arasınav
8. **Geçici Tablolar, Kontrol Yapıları**
9. **Stored Procedure**
10. Fonksiyonlar
11. Tetikleyiciler
12. Transaction Kavramları ve Yedekleme
13. Kullanıcı Türleri ve Kullanıcı Yönetimi
14. No-SQL Veri Tabanları

Örnek Sorgular

- Hastaların isimleri, soyisimleri, hastaların yakalandıkları virüslerin isimleri ve virüslerin türlerini listeyeleyen bir view yazınız.

Örnek Sorgular

- Hastaların isimleri, soyisimleri, hastaların yakalandıkları virüslerin isimleri ve virüslerin türlerini listeyeleyen bir view yazınız.
- Çözüm:

```
CREATE VIEW view_HastaVirus AS
  SELECT HastaAd=TH.Ad,
  HastaSoyad=TH.Soyad,
  VirusAd=TV.Ad,
  VirusTur=TVT.Ad
FROM tbl_Hasta TH INNER JOIN tbl_HastaVirus THV ON TH.id=THV.HastaId
INNER JOIN tbl_Virus TV  ON TV.id=THV.VirusId
INNER JOIN tbl_VirusTur TVT ON TV.virusTurId=TVT.id
```

Örnek Sorgular

- Doktorların isimleri, soyisimleri, baktıkları hasta sayılarını listeyen bir view yazınız.

Örnek Sorgular

- Doktorların isimleri, soyisimleri, baktıkları hasta sayılarını listeyen bir view yazınız.
- Çözüm:

```
CREATE VIEW view_DoktorHastaSayisi AS
SELECT DoktorAd=TP.Ad,
DoktorSoyad=TP.Soyad,
HastaSayisi=COUNT(*)
FROM tbl_Personel TP LEFT JOIN tbl_HastaMuayene THM
ON TP.id=THM.personelId
GROUP BY TP.AD,TP.soyad
```

Temp Table

- Sistem üzerinde geçici olarak saklanmak istenen tablolar
- Mevcut tabloların bir alt kümesi olarak saklanabilir.
- Özellikle çok sayıda kaydı olan ve bu kayıtların küçük bir kısmı üzerinde sürekli olarak etkileşime girilmesi gerektiği durumlarda oluşturulur.
- Alt kümeyi oluşturan sorguların tekrar tekrar üretilmesi yerine bir defa üretilip, geçici tabloya aktarılır.

Temp Table

- **Nerede:** System Databases > tempdb > Temporary Tables
- **Oluşturma**:
  - Yöntem 1:

```
CREATE TABLE #temp_Personel_kodOlusturma
(
   id int,
   adi NVARCHAR(50)
)
```

  - Yöntem 2:

```
SELECT * INTO #tablo_Adi FROM tbl_Personel
```

- **Kullanımı**:
- select * from tempdb.#temp_Personel

Temp Table

- Silinmesi için 2 yöntem vardır.
  - Otomatik Silme
    - Tabloyu oluşturan bağlantı kapatıldığında geçici tablo otomatik olarak silinir.
    - Eğer geçici tablo üzerinde işlem yapan başka bir sorgu varsa önce bu sorgu tamamlanır ve sonra silinir.
  - Kod ile Silme
    - Drop table #tablo_Adi

ROW_NUMBER() ifadesi

```
SELECT
ROW_NUMBER() OVER (ORDER BY tbl_PersonelTur.id ASC) Id,
Gorevi=tbl_personelTur.Ad,
PersonelAdi=tbl_personel.Ad+ ' '+tbl_personel.Soyad
FROM tbl_personel
INNER JOIN tbl_PersonelTur ON tbl_PersonelTur.id=personelTurId
```

Değişken Tanımları

- **Declare** @parameterName **ParameterTYpe**
- SET @parameterName=Value

  - DECLARE @param CHAR(10)
    SET@param=GETDATE()
    SELECT @param

  - DECLARE @deger INT
    SELECT @deger =1
    SELECT @deger

CASE-WHEN Yapısı

- DECLARE @deger INT

SET @deger=1

SELECT

```
        CASE @deger
                WHEN 1 THEN 'bir'
                WHEN 2 THEN 'iki'
        END
```

CASE-WHEN Yapısı

```
SELECT
        Ad,
        CASE
                WHEN adres IS NULL THEN 'Yok'
                ELSE adres
        END
FROM tbl_Personel
```

CASE-WHEN Yapısı

```
SELECT
        TPT.Ad,
        PersonelSayisi=COUNT(TP.id),
        CASE
                WHEN COUNT(tp.id)<=2  THEN 'Personel Alinmali'
                ELSE 'Yeterli Personel Var'
        END AS PersonelDurumu
FROM tbl_Personel AS TP
        RIGHT JOIN tbl_PersonelTur AS TPT ON personelTurId=TPT.id
        GROUP BY TPT.Ad
```

CASE-WHEN Yapısı

```
SELECT
        TPT.Ad,
        PersonelSayisi=COUNT(TP.id),
        CASE
                WHEN COUNT(tp.id)<=1  THEN 'Kesin Personel Alinmali'
                WHEN COUNT(tp.id)<5 THEN 'Personel Alinsa İyi olur '
                ELSE 'Yeterli Personel Var'
        END AS PersonelDurumu
FROM tbl_Personel AS TP  RIGHT JOIN tbl_PersonelTur AS TPT ON personelTurId=TPT.id
        GROUP BY TPT.Ad
```

IF –ELSE Yapısı

```
DECLARE @id int
SET @id=1
DECLARE @param AS VARCHAR(50)
        IF (@id=1)
                SET @param='BİR'
        ELSE
                SET @param='İKİ'
SELECT @param
```

IF –ELSE IF Yapısı

```
DECLARE @id int
SET @id=3
DECLARE @param AS VARCHAR(50)
        IF (@id=1)
                SET @param='BİR'
        ELSE IF(@id=2)
                SET @param='İKİ'
        ELSE
                SET @param='DEFAULT'
SELECT @param
```

IF –ELSE IF Yapısı

```
DECLARE @Tarih DATETIME
SET @Tarih='2020-02-01'---CONVERT
--SET
@Tarih=CONVERT(Datetime,'01.12.2019',104)
DECLARE @Ad VARCHAR (7)
DECLARE @AyNo TINYINT
SET @AyNo = MONTH(@Tarih)

IF @AyNo = 1 SET @Ad = 'Ocak‘
ELSE IF @AyNo = 2 SET @Ad = ‘Şubat‘
ELSE IF @AyNo = 3 SET @Ad = 'Mart'
ELSE IF @AyNo = 4 SET @Ad = 'Nisan'
ELSE IF @AyNo = 5 SET @Ad = 'Mayıs'
ELSE IF @AyNo = 6 SET @Ad = 'Haziran'
ELSE IF @AyNo = 7 SET @Ad = 'Temmuz'
ELSE IF @AyNo = 8 SET @Ad = 'Ağustos'
ELSE IF @AyNo = 9 SET @Ad = 'Eylül'
ELSE IF @AyNo = 10 SET @Ad = 'Ekim'
ELSE IF @AyNo = 11 SET @Ad = 'Kasım'
ELSE IF @AyNo = 12 SET @Ad = 'Aralık'
SELECT @Ad
```

WHILE Yapısı

```
DECLARE @x int

SET @x=1

WHILE (@x<3) BEGIN

        SET @x=@x+1

END

SELECT @x
```

WHILE Yapısı

```
DECLARE @x int
SET @x=1
WHILE (@x<3) BEGIN
        SET @x=@x+1
        IF (@x=3)
                SELECT 'EXIT'
END
SELECT @x
```

TRY-CATCH Yapısı

```
BEGIN TRY
        PRINT 3/0;
END TRY
BEGIN CATCH
        PRINT 'Catch bloğunun içi';
        PRINT ERROR_NUMBER();
        PRINT ERROR_MESSAGE();
        PRINT ERROR_NUMBER();
END CATCH
PRINT 'Catch bloğu bittikten sonra'
PRINT ERROR_NUMBER()
```

CURSOR İfadesi

```
DECLARE @name AS nvarchar(50)
DECLARE @currentPersonCount AS INT
DECLARE @index AS INT
SET @index=0
DECLARE @sonuc AS VARCHAR(50)
DECLARE personelControl Scroll CURSOR FOR
SELECT
                TPT.Ad,
                PersonelSayisi=COUNT(TP.id)
FROM tbl_Personel TP
                RIGHT JOIN tbl_PersonelTur TPT  ON TPT.id=PersonelTurId
                GROUP BY TPT.Ad
```

```
OPEN personelControl
FETCH personelControl INTO @name,@currentPersonCount
WHILE (@@Fetch_Status<>-1)
Begin
   if (@currentPersonCount <2)
                SELECT @sonuc=@name+'-Alýnmalý'
   Fetch personelControl into @name,@currentPersonCount
   SET @index=@index+1
    End
 Close personelControl
DeAllocate personelControl
 Select @sonuc as sonuc
SELECT CAST(@index AS NVARCHAR) +' ' +'SATIR gezildi'
```

Stored Procedure(SP)

- SP belirli bir işlevi, görevi yerine getirmek için yazılan SQL ifadeleri
- Sık kullanılan yapıların SP olarak tek bir defa yazılması ve kullanılmak istendiği yerde sadece çağrılması (Tekrar tekrar kullanılması)
- İlk çalıştığında derlenir, sonrasında sadece işletilir derleme ihtiyacına gerek kalmaz
- Server tarafında saklananır ve parametre alabilir
- Verinin saklanma biçiminin ve/veya tabloların tasarımının saklanması

Stored Procedure Türleri

- **Extended Stored Procedure:** Genellikle *.dll şeklinde prosedürlerdir.
- **CLR Stored Procedure:** SQL Server 2005 sonrasında CLR ortamında herhangi bir dili kullanarak kodlanan SP'lerdir.
- **System Stored Procedure:** Genellikle sp_ ön ekiyle başlarlar ve hepsi master veri tabanında tutulan SP'lerdir.
- **Kullanıcı Tanımlı SP:** Kullanıcıların tanımladığı SP'lerdir.

Stored Procedure

- Genel Yapısı
  - CREATE PROCEDURE/PROC prosedur_adı

    Parametre seçenekleri

    AS

    BEGIN

    T-SQL ifadeleri

    END
  - Çalıştırma Seçeneği:
    - EXEC prosedur_adı

Stored Procedure

```
CREATE PROC  sp_getPersonelBilgileri AS BEGIN
 SELECT TP.ad,Soyad,
gorevi=
            CASE
                        WHEN TP.personelTurId IS NULL THEN '-'
                        ELSE TPT.Ad
            END
FROM tbl_Personel TP LEFT JOIN
tbl_PersonelTur TPT ON TP.personelTurId=TPT.id
 END
```

Çalışma: EXEC sp_getPersonelBilgileri

Parametre Alan Store Procedure

```
CREATE PROCEDURE sp_DoktorGore_HastaSayilari
  @doktorAdi   NVARCHAR(50)
AS   BEGIN
 SELECT DoktorAd=TP.Ad,
    DoktorSoyad=TP.Soyad,
    HastaSayisi=COUNT(*)
    FROM tbl_Personel TP
 LEFT JOIN tbl_HastaMuayene THM
 ON TP.id=THM.personelId
 WHERE TP.Ad LIKE '%'+@doktorAdi+'%'
 GROUP BY TP.AD,TP.soyad
END
```

Çalışma:

```
EXEC sp_DoktorGore_HastaSayilari 'Ali'
EXEC sp_DoktorGore_HastaSayilari @doktorAdi ='asli‘
```

Parametre Döndüren SP

```
CREATE PROCEDURE sp_ToplamPersonelSayisi
@sayi AS INT OUT
AS
BEGIN
        SELECT @sayi=COUNT(*) FROM tbl_Personel
END
```

Çalışma:

```
DECLARE @Personelsayisi AS INT
EXEC sp_ToplamPersonelSayisi @Personelsayisi OUT
SELECT @Personelsayisi
```

yada

```
DECLARE @Personelsayisi AS INT
EXEC sp_ToplamPersonelSayisi @sayi=@Personelsayisi OUT
SELECT @Personelsayisi
```

SP-INSERT

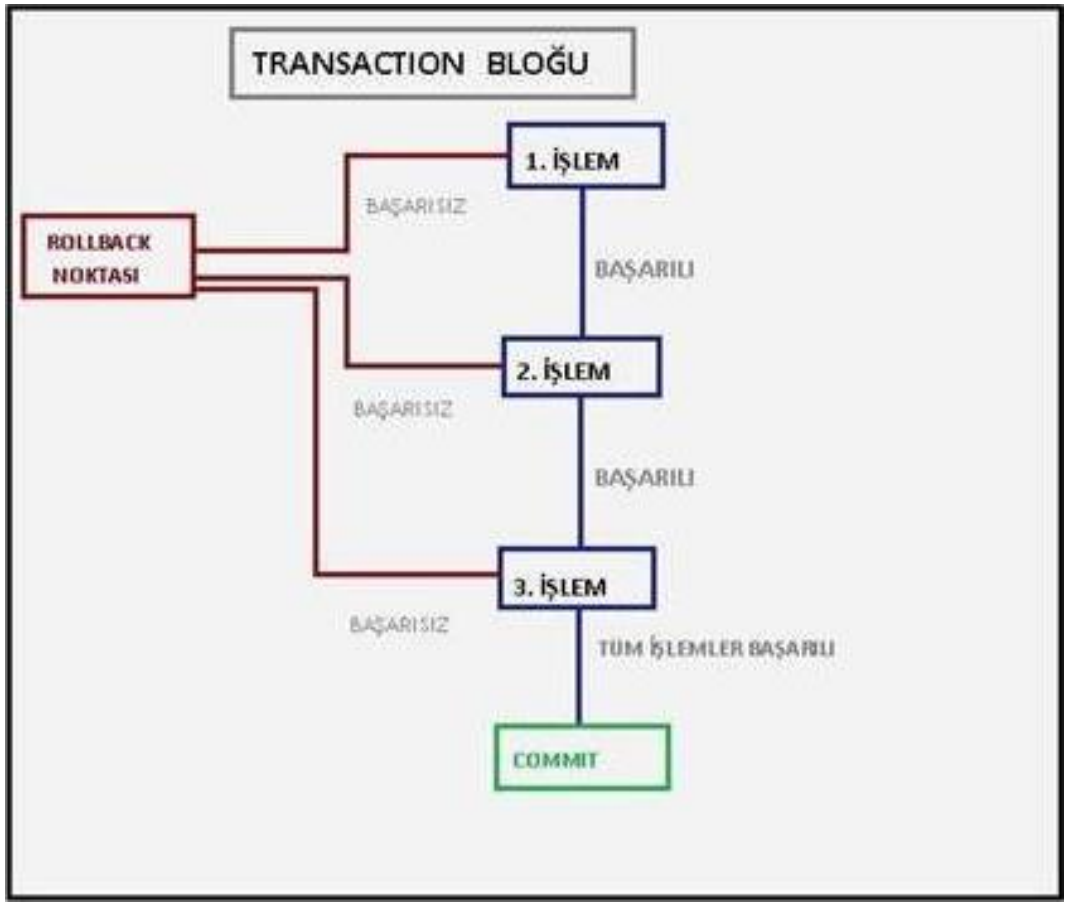
```
CREATE PROCEDURE sp_Add_HastaMuayene

@HastaId AS INT, @PerosnelId AS INT,

@tarih AS DATE

AS  BEGIN

INSERT  INTO tbl_HastaMuayene values (@HastaId,@PerosnelId,

NULL,@tarih)

END
```

Çalışma: SET DATEFORMAT DMY;

```
EXEC sp_Add_HastaMuayene  @HastaId =3 , @PerosnelId =3, @tarih='23.11.2022'
```

Transaction

- Transaction en küçük işlem bloğu
- Transaction;
  - Ya bütün işlemleri gerçekleştirir ya da hiçbirini gerçekleştirmez.
- BEGIN ile başlar
- ROLLBACK : Tüm işlemleri geri alır
- COMMIT: Tüm işlemlerin başarılı şekilde yapıldığı düşünülür.
- ACID özelliği



SP-Transaction

```
CREATE PROCEDURE sp_Add_HastaMuayeneV2

@HastaId AS INT, @PerosnelId AS INT, @tarih AS DATE

AS  BEGIN

BEGIN TRANSACTION

INSERT  INTO tbl_HastaMuayene values
(@HastaId,@PerosnelId, NULL,@tarih)

IF (@@error <> 0)

BEGIN
  SELECT 'ROLLBACK'
  ROLLBACK TRANSACTION

END

COMMIT TRANSACTION

END
```

Çalışma:

```
SET DATEFORMAT DMY;

EXEC sp_Add_HastaMuayeneV2  @HastaId =3 , @PerosnelId
=3, @tarih='23.11.2022'
```

SP-Transaction

```
CREATE PROCEDURE sp_Add_HastaMuayeneV3

 @HastaId AS INT, @PerosnelId AS INT, @tarih AS DATE,

@sonuc AS INT OUT
AS BEGIN
BEGIN TRANSACTION
  INSERT  INTO tbl_HastaMuayene values
            (@HastaId,@PerosnelId, NULL,@tarih)

IF (@@error <> 0)
            BEGIN
                        SELECT @sonuc=0
                        SELECT 'ROLLBACK'
                        ROLLBACK TRANSACTION
            END
ELSE
            SELECT @sonuc=1
            COMMIT TRANSACTION
END
```

Çalışma:

```
DECLARE @islemSonucu AS INT

SET DATEFORMAT DMY;

EXEC sp_Add_HastaMuayeneV3 @HastaId =3 , @PerosnelId =3,

@tarih='23.11.2022', @sonuc=@islemSonucu OUT
SELECT @islemSonucu
```

ACID(**A**tomicity, **C**onsistency, **I**solation, **D**urability)

- **Atomicty (Bölünmezlik):** Bir transaction bloğu yarım kalamaz. Yarım kalan transaction bloğu veri tutarsızlığına neden olur. Ya tüm işlemler gerçekleştirilir, ya da transaction başlangıcına geri döner.
- **Consistency (Tutarlılık):** Transaction veri tutarlılığı sağlamalıdır. Yani bir transaction içerisinde güncelleme işlemi gerçekleştiyse ve ya kalan tüm işlemler de gerçekleşmeli ya da güncelle işlemi de geri alınmalıdır.
- **Isolation (İzolasyon):** Bir transaction tarafından gerçekleştirilen değişiklikler tamamlanmadan bir başka transaction tarafından görülememelsi gerekir. Yani her transaction ayrı ayrı işlenmelidir.
- **Durability (Dayanıklılık):** Transaction'lar veri üzerinde karmaşık işlemler gerçekleştirirken verinin bütününü güvence altına almalıdır. Bu sebeple transaction hatalara karşı dayanıklı olmalıdır.