Esra RAMAZAN

Veritabanı Yönetim Sistemleri

Danışman

Öğr. Gör. Cevahir PARLAK

2014

Projede bölüm, doktor, hasta bilgileri, randevu bilgileri tutulacaktır. Bu proje için bir veritabanı istenmektedir. Veritabanı iş emirleri aşağıdaki gibidir.

1. Bölüm bilgileri olmalıdır. Bölüm adı, Bölüm numarası ve liste numaraları.
   Bolum(nu, Bolum_id, Bolum_adi)
2. Doktor bilgileri olmalıdır. Doktor adı ve soyadı, Doktor numarası, liste numarası, cinsiyeti, bölüm id'si.
   Doktor(nu, Doktor_id, Bolum_id, adi, cinsiyet)
3. Hasta kayıt bilgileri olmalıdır. Hastanın adı, soyadı, Hasta id'si, liste numarası, cinsiyeti, bölüm id'si, doktor id'si,tc, randevu tarihi, cep numarası,
   Hasta_kayit(nu, Doktor_id, Bolum_id, adi, cinsiyet, soyadi, tc, randevu_tarihi, cep_nu,)
4. Çalışır bağlantı tablosu yapılmalı ve bölüm ile doktor id bilgileri içermelidir.
   Calisir(Bolum_id, Doktor_id)
5. Randuvu bağlantı tablosu da yapılıp içine tc, bolum_id, doktor id ve hasta id'si olmalıdır. Bunların birleşimi randevuyu oluşturmalıdır.
6. Prosedür ve trigger kullanımı şarttır. Bölüm silindiğinde otomatik olarak bölüme ait her bilgi kaldırılmalıdır.
7. Doktor kaldırıldığında bölüme dokunulmaksızın hasta kaydı randevusu ve çalışır bağlantı tablosundan doktora ait bilgiler silinmelidir.
8. Hasta randevuyu iptal ettiğinde randevudan otomatik olarak kaydı silinmelidir.
9. Randevu alındığında hasta bilgileri ile otomatik olarak randevu tablosuna da veriler eklenmelidir. Ayrıca kullanıcı randevu tablosu içinde veri girilmemeli.
10. Doktor eklendiğinde çalışır tablosuna otomatik bilgileri eklenmelidir.
11. Güncellemeler otomasyon üzerinden olmalıdır.
12. Bir bölümde birden fazla doktor çalışabilir ancak bir doktor yalnızca bir bölümde çalışabilmektedir.
    Burada Bire çok ilişki vardır.
13. Bir hasta birden fazla bölümde ve birden fazla doktora muayene olabilir.
    Burada Çoka çok ilişki vardır.

```
Create database Hastahane
```

(Nu alanları sıra numarası içindir.)

```
Create table Bolum
(
nu int identity(1,1),
Bolum_id int primary key not null,
Bolum_adi varchar(40) not null
)

Create table Doktor
(
nu int identity(1,1),
Doktor_id int primary key not null,
Bolum_id int foreign key references Bolum(Bolum_id) not null,
Ad varchar(50) not null,
Cinsiyet varchar(5) not null
)

Create table Calisir
(
Doktor_id int foreign key references Doktor(Doktor_id) not
null,
Bolum_id int foreign key references Bolum(Bolum_id) not null
)

Create table Hasta_kayit
(
Nu int identity(1,1),
Hasta_id int primary key not null,
Tc bigint not null,
Adi varchar(20) not null,
Soyadi varchar(30) not null,
Cinsiyet varchar(5) not null,
Randevu_tarihi smalldatetime not null,
Cep_Nu bigint not null,
Doktor_id int foreign key references Doktor(Doktor_id) not
null,
Bolum_id int foreign key references Bolum(Bolum_id) not null
)
```

```
Create table Randevu
(
Hasta_id int foreign key references Hasta_kayit(Hasta_id) not
null,
Doktor_id int foreign key references Doktor(Doktor_id) not
null,
Bolum_id int foreign key references Bolum(Bolum_id) not null,
tc bigint not null
)
```

Randevu silinirse otomatik olarak hastanın bilgileride silinir ve hastanın işlemi yapması kolaylaştırılmış olur.

```
go
create proc sp_RandevuSil
@tc bigint
as
     delete Randevu where tc=@tc
     delete Hasta_kayit where Tc=@tc
go
```

Doktor silinirse otomatik olarak altındaki tüm veriler silinir böylelikle tek tek yönetici silme işlemi yapmaz.

```
create proc sp_DoktorSil
@Doktor_id int
as
     delete Randevu where Doktor_id=@Doktor_id
     delete Hasta_kayit where Doktor_id=@Doktor_id
     delete Calisir where Doktor_id=@Doktor_id
     delete Doktor where Doktor_id=@Doktor_id
go
```

Bölüm silinirse otomatik olarak bölüme bağlı tüm veriler silinir böylelikle yönetici tek tek silme işlemi yapmaz.

```
create proc sp_BolumSil
@Bolum_id int
as
     delete Randevu where Bolum_id=@Bolum_id
     delete Hasta_kayit where Bolum_id=@Bolum_id
     delete Calisir where Bolum_id=@Bolum_id
     delete Doktor where Bolum_id=@Bolum_id
     delete Bolum where Bolum_id=@Bolum_id
go
```

Yönetim doktor bilgilerini girdikten sonra otomatik olarak çalışır bağlantı tablosunada veriler gidiyor böylelikle yönetici daha hızlı ve kolay çalışıyor.

```
go
create trigger tri_DoktorEkle on Doktor
for insert
as
     declare @bolumID int
     declare @DoktorID int
     declare @Ad varchar(20)
     declare @Cinsiyet varchar(5)
select
@DoktorID=Doktor_id,@bolumID=Bolum_id,@Ad=Ad,@Cinsiyet=Cinsiye
t from inserted
begin

     insert into calisir values (@DoktorID,@bolumID)
end
```

Hasta kendi bilgilerini girdikten sonra otomatik olarak randevu bağlantı tablosunada veriler gidiyor böylelikle hasta hem kaydını hem randevuyu eklemek durumunda kalmıyor.

```
go
create trigger tri_RandevuEkle on Hasta_kayit
for insert
as
     declare @Hasta_Nu int
     declare @Tc bigint
     declare @Adi varchar(20)
     declare @Soyadi varchar(30)
     declare @Cinsiyet varchar(5)
     declare @Randevu_tarihi smalldatetime
     declare @Cep_Nu bigint
     declare @Doktor_id int
     declare @Bolum_id int
select
@Hasta_Nu=Hasta_id,@Tc=Tc,@Adi=Adi,@Soyadi=Soyadi,@Cinsiyet=Ci
nsiyet,@Randevu_tarihi=Randevu_tarihi,@Cep_Nu=Cep_Nu,@Doktor_i
d=Doktor_id,@Bolum_id=Bolum_id from inserted
begin

     insert into Randevu values
(@Hasta_Nu,@Doktor_id,@Bolum_id,@Tc)
end
```