4. Bölüm

- DDL -
VERİ TANIMLAMA DİLİ

DDL (Data Definition Language – Veri Tanımlama Dili) : Bu kategorideki SQL komutları ile veritabanları, tablo, görünüm ve indekslerin yaratılması, silinmesi ve değişikliklerin yapılması gibi işlemler yapılabilmektedir. Örnek : Create, Drop, Alter vb.

4.1. Oluşturma Deyimi (Create)

Create deyimi tablo, indeks ve view gibi veritabanı nesnesi oluşturmada kullanılmaktadır.

4.1.1. Tablo Oluşturma (Create Table)

```
Create Table    Kimlik
   (    Numara        Char(9)  ,
        Ad            Char(15) ,
        Soyad         Char(15) ,
        Telefon       Char(13) ,
        Adres         Char(50)
   )
```

4.1.2. Kısıtlamalar (Constraints)

Kısıtlamalar sayesinde verilerin istenen şartlarda girişleri sağlanmış olmaktadır. Kısıtlama yapılmadan kullanılan alanlar için Default değerler geçerlidir. Tablo yapısında mevcut olan, ancak ekleme durumunda değeri atanmayan alanlar için Default değerler geçerlidir.

İki tür kısıtlama mevcuttur. Bunlar :

- **Tablo Kısıtlamaları** : Tablonun bir veya daha çok sütunu için yapılan kısıtlamalardır. Bu tür kısıtlamalar tabloya ait bütün sütun tanımlamaları bittikten sonra yapılır.
- **Sütun Kısıtlamaları** : Sadece belirtilen sütun için kısıtlama getirilir. Kısıtlama işlemi alan isminin hemen yanında yapılır.

Kısıtlamalar	İşleyişi
Not Null	Veri girişinde Null değerler engellenir. (Sadece Sütun sınırlaması)
Unique	Girilen verinin tabloda tek olması sağlanır.
Primary Key	Birincil Anahtar özelliği verir. Tabloda tek olmasını sağlar. Birden çok alan için Tablo Kısıtlaması kullanılır.
Foreign Key	Yabancı anahtar özelliği verilir.

Aşağıdaki örnekte tablo ve sütun kısıtlamaları gösterilmektedir.

```
Create Table Kimlik
       ( Numara      Char(9)    Primary Key, <<<  Sütun Kısıtlaması
         Ad          Char(15),
         Soyad       Char(15),
         Telefon     Char(13) ,
         Adres       Char(50)

        Unique       (Ad,Soyad)              <<< Tablo Kısıtlaması
    )
```

Örnek 1:

```
Create Table  Dersler
(       DersKodu        Char(7)     Primary Key,
        DersAdi         Char(20),
        Teori           SmallInt ,
        Pratik          SmallInt ,
        Kredi           SmallInt
)
```

DersKodu alanına sütun kısıtlaması ile Birincil Anahtar özelliği verilmiştir.

Örnek 2:

```
Create Table  Notlar
(       Numara          Char(9) ,
        DersKodu        Char(4) ,
        Donem           Char(9) ,
        Vize            Integer   Not Null,
        Final           Integer ,
        Ortalama        Real

        Primary Key     (Numara, DersKodu, Donem)
)
```

Yukarıda Numara, DersKodu, Donem ve Vize alanlarına veri girişi boş geçilemeyecektir. Numara, Donem ve DersKodu alanlarına da birincil anahtar özelliği verilmiştir. Tabloya fiziksel olarak kayıt girişleri bu alanlar dikkate alınarak yapılacaktır. Gereksiz yere fazla miktarda Primary Key kullanılması performansın düşmesine yol açacaktır.

Örnek 3:

```
Create Table    Notlar
(       Numara          Char(9) ,
        Donem           Char(9) ,
        DersKodu        Char(4) ,
        Vize            Integer ,
        Final           Integer ,
        Ortalama        Real

        Unique          (Numara, Donem, DersKodu) ,
        Primary Key     (Numara, Donem, DersKodu)
)
```

Soru : Yukarıda Unique ve Primary Key kullanımlarından hangisinin olmaması sonucu etkilemez?

Açıklama : Yukarıdaki kullanımda Primary ve Unique kullanımı aynı şey değildir.

"Primary Key (Numara, Donem, DersKodu)" kullanımında geçerli olan özellikler aşağıda verilmektedir :

- Numara, Donem ve DersKodu alanlarına göre Birincil Anahtar (Fiziksel indeks) kullanımı (Bu alanlara göre sıralı olmayı sağlar)
- Numara, Donem ve DersKodu alanların hepsi aynı olan ikinci bir kayıt yapılmasını engelleme
- Numara, Donem ve DersKodu alanlarına Null girişi engelleme (Not Null özelliği verme)

"Unique (Numara, Donem, DersKodu)" kullanımında geçerli olan özellikler aşağıda verilmektedir :

- Numara, Donem ve DersKodu alanların hepsi aynı olan ikinci bir kayıt yapılmasını engelleme
- Bu alanlar kombinasyonu için yalnız bir defa olacak şekilde Null giriş yapılabilir

Cevap : Bu durumda **Unique** olmadan da sonuç aynı olur.

Örnek 4: **Default değer kullanımı.**

```
Create Table Kimlik
        (
                Numara          Char(9)         Primary Key,
                Ad              Char (15),
                Soyad           Char(15),
                MedeniHal       Char (10)       Default 'Bekar'
        )
```

Tablodaki alanın varsayılan değerinin belirlenmesinde Defult kullanılır. Burada MedeniHal alanı için varsayılan değer '**Bekar**' olarak tanımlanmaktadır.

4.1.3. İndeks Oluşturma (Create Index)

İndeksler kullanılarak sorgulama işlevleri hızlandırılabilmektedir. Ancak çok sayıda indeks kullanılması da sorunlar yaratabilmektedir. Bir tablo için maksimum 5 ya da 6 tane indeks tanımlanabilir. Ayrıca Primary Key ve Foreign Key alanlar için İndeks tanımı yapılmamalıdır.

Create Index İndeks_Adı **on** Tablo_Adı (Sütun1 İndeks_Yönü, Sütun2 İndeks_Yönü, ...)

İndeks_Adı	**:** Oluşturulan indekse verilecek isim
Tablo_adı	: İndeksin geçerli olduğu **tablo** ya da **görünüm** adı
Sütun	**:** İndeksin tabloda hangi alan ya da alanlara göre yapılacağı belirtilir.
İndeks_Yönü	**:** Asc (default) artan sıralama, Desc azalan sıralama verir.

Örnek 1: **Create Index** ndx_OgrNo **on** Kimlik (OgrNo)

Kimlik adlı tabloda OgrNo alanına göre, ndx_OgrNo adında indeks oluşturulur.

Örnek 2: **Create Index** ndx_Isim **on** Kimlik (Ad , Soyad)

Kimlik adlı tabloda Ad ve Soyad alanlarına göre, ndx_Isim adında indeks oluşturulur. Ad aynı ise Soyada göre sıralama yapılır.

Örnek 3: **Create Index** ndx_Isim **on** Kimlik (Ad **Asc**, Soyad **Desc**)
Kimlik adlı tabloda Ad alanına göre artan, Soyad alanına göre azalan olacak şekilde, ndx_Isim adında indeks oluşturulur.

Örnek 4: **Create Unique Index ndx_OgrNo on** Notlar **(OgrNo)**

Notlar adlı tabloda OgrNo alanına göre, ndx_OgrNo adında ve tekrarlı kayıtlara izin verilmeyen indeks oluşturulur.

4.1.4. Görünüm Oluşturma (View)

Mevcut tablo ya da tabloların sadece istenilen sütunlarının veya belirtilen şartlara uyan kayıtlarının görülüp üzerinde işlemlerin yapılmasına izin verilmesi amacıyla kullanılmaktadır. Bu sayede tablolara ait değişik View'ler oluşturularak, kullanıcıların belli yetkiler dahilinde verilere erişmeleri sağlanabilmektedir. Bu verilerin güvenliği açısından son derece önemlidir. Ayrıca karmaşık sorgulardan elde edilen görünümlerin kullanılması uygulamada kolaylık sağlamaktadır.

View yapıları sadece görüntü amaçlıdır. Kayıtlar veritabanında fiziksel olarak sadece tablolarda tutulmaktadır.

Görünümler üzerinde kayıt ekleme, silme ve güncelleme işlemleri aynı tablolarda olduğu gibi yapılabilmektedir. Ancak görünümlerde oluşturulma şartına bağlı olarak ekleme yapılabilmektedir.

```
Create View    View_Adı ( View_Sütun1, View_Sütun2 … )
As
SQL_Cümlesi
```

View_Adı : Oluşturulacak View için verilecek ad.
View_Sütun : Oluşturulan View'da sütunların alacağı adlar.
SQL_Cümlesi : SQL değimlerinden oluşan ifade.

Not : View oluşturulurken Order By, Compute By ve Into kullanılamamaktadır.

Örnek 1:

```
Create View V_Ogrenci (Ogrenci_No, Ders_Kodu, Ortalama )
As
Select  Numara, DersKodu, Ortalama From Ogrenci
```

Numara, DersKodu, Vize, Final ve Ortalama alanlarından oluşan 'Ogrenci' adlı tablodan 'V_Ogrenci' adında sadece Numara, DersKodu ve Ortalama alanlarını içeren bir view oluşturulmuştur.

Örnek 2:

```
Create View V_Ogrenci (Numara, Ortalama )
As
Select  Numara, Ortalama From Ogrenci
Where  DersKodu = '4510207'
```

'4510207' kodlu derse ait sadece Numara ve Ortalama alan değerlerini içeren bir View oluşturulmuştur. Bu ders kodundan başka derslere ait kayıtlar görülemez.

Örnek 3:

```
Create View V_Ortalama (OgrNo, OgrAd, OgrSoyad, DersKodu, Ortalama)
As
Select   K.Numara, K.Ad,  K.Soyad, N.DersKodu, N.Ortalama
From     Kimlik as K,   Notlar as N
Where    ( K.Numara = N.Numara )  And  ( N.DersKodu = '4510207' )
```

'Kimlik' ve 'Notlar' adlı iki ayrı tablodan Numara alanına göre eşit bağlantı yapılarak sadece '4510207' kodlu ders için ilgili alanlar alınarak 'V_Ortalama' adında bir View oluşturulmuştur.

Örnek 4:

```
Insert Into   V_Ortalama (OgrNo, OgrAd, OgrSoyad, DersKodu, Ortalama)
Values        ('094510006', 'ZEYNEP', 'DİŞLİTAŞ', '4510207', 65)
```

'V_Ortalama' adında bir görünüme kayıt eklenmek istenmektedir. Ancak kayıt çok sayıda tabloyu içerdiğinden aşağıdaki gibi bir mesaj verilerek kayıt engellenmektedir.

```
View or function 'V_Ortalama' is not updatable because the modification affects multiple base tables.
```

NOT: View tek bir tablo ile ilişkili ise; bu durumda Null giriş izinleri de dikkate alınarak Insert Into kullanımlarına izin verilebilir.

Örnek 5:

```
Create View v_Notlar (Numara, DersKodu, Ogretim_Yili, Harf_Notu)
as
Select okulno, yoptikkod, ogretimyili, harfsonuc From Notlar
Where  Okulno Like '__4526%'
```

Örnek 6: **Select * From V_Ortalama**

Yukarıdaki sorguda; V_Ortalama adlı view içerisindeki tüm kayıtlar elde edilir. Aynı normal bir tablo gibi kullanılabilir.

Örnek 7:

```
Select * From v_Notlar
Where Derskodu like '4526209%'
```

Örnek 8:

```
Drop view v_Notlar
```