








## Hedefler

Bu üniteyi çalıştıktan sonra;

-  Veritabanlarını öğrenmek,
-  Temel SQL Komutlarını öğrenmek,
-  ADO.NET ve Veritabanı Erişim Araçlarını öğrenmek,
-  Veri Sağlayıcıları öğrenmek,
-  DataGridView Kontrolünü öğrenmek,

### İçindekiler

- Veritabanları,
- Temel SQL Komutları,
- ADO.NET ve Veritabanı Erişim Araçları,
- Veri Sağlayıcılar,
- DataGridView Kontrolü.

## VERİTABANLARI

Veritabanı, istenilen bilgi yada bilgi parçacıklarının belirli bir düzende depolanması için kullanılan düzenli bilgi kümeleridir. Microsoft Access, Microsoft FoxPro, Oracle ve SQL Server gibi çeşitli veritabanı programları kullanılarak güçlü veritabanları oluşturulabilir. Veritabanı uygulaması, bir veritabanının alan ve kayıtlarını alarak bunları kullanıcılar için anlamlı bir biçimde görüntüleyen bir programdır. Kayıtları aramak, silmek, yazdırmak, eklemek bir veritabanı uygulamasının temel komutlarıdır.

Elimizdeki bilgileri çeşitli bilgi kanallarından kolayca aktarabilmeli ve bu kanallardan istediğimiz bilgileri alabilmeliyiz. İşlerimizde ve günlük yaşantımızda interneti çok sık kullanmaktayız. Bilgilerimiz internet üzerinden text olarak gidip geliyor. Tüm bu gelişmeler sonunda verilerimiz için ortak bir tanımlama dili oluşturuldu. Bu dil hepimizin bildiği gibi XML.

İnternet ortamındaki bu gelişmelerden sonra Microsoft kendi veri erişimini endüstri standartlarına uygun ve internet ortamına kolay taşınabilir bir şekilde değiştirdi ve ADO.NET ortaya çıktı. ADO.NET veri iletişimi üzerinde veriler XML olarak tutuluyor.

ADO.NET, OLEDB uyumlu veritabanları (Access, Oracle, FoxPro vs.), SQL Server veritabanları, ilişkisel olmayan veritabanları ve XML belgeleri de olmak üzere birçok veri dosyasına erişim sağlar.

Veritabanları üzerinde işlem yapmak için yararlanabileceğimiz veritabanı nesneleri şunlardır :

- **Connection** : Veritabanı ile bağlantı kurmak için kullanılır.
- **Command** : Veritabanı işlemleri ile ilgili sorgular için kullanılır.
- **DataReader** : Veritabanı içerisindeki verileri okumak için kullanılır.
- **DataSet** : Veritabanı içerisindeki verileri okuma ve yazma için kullanılır.
- **DataAdapter** : Veritabanı bağlantısı olmasa bile veriler üzerinde işlemler gerçekleştirilebilir.

## Temel SQL Komutları

Bu bölümde temel SQL komutları ile veritabanını sorgulayarak veri alma, sıralama, güncelleştirme, silme gibi işlemlerinin nasıl yapıldığını inceleyeceğiz.

### SELECT İfadesi

#### SELECT İfadesi

**SELECT** ifadesi veritabanından bilgi seçmek için kullanılır. Genel ifadesi aşağıdaki şekildedir.

**SELECT** Sütun\_ismi\_1, Sütun\_ismi2, .... **FROM** Tablo\_ismi **WHERE** Koşul **ORDER BY** Sıralanacak\_sütun [ASC / DESC]

**Temel SELECT İfadesi** : SELECT ifadesinin en basit şekli,

**SELECT \* FROM** Tablo\_ismi

Buradaki (\*) tablodaki tüm sütunları (alan) seçecektir.

**Belirli Sütunların Seçilmesi** : Tablodan istenilen sütunlar seçilebilir.

**SELECT** Sütun\_ismi\_1, Sütun\_ismi\_2, .... **FROM** Tablo\_ismi

**WHERE İfadesi** : Koşulu sağlayan bilgilerin tablodan seçilmesini sağlar.

**SELECT** sütun\_ismi\_1, Sütun\_ismi\_2, .... **FROM** Tablo\_ismi **WHERE** Koşul

Koşullarda kullanılan operatörler:

Karşılaştırma Operatörleri	=, <>, <, <=, >, >=
Mantıksal Operatörler	NOT, OR, AND
Alfabetik Operatörler	LIKE, NOT LIKE
Değer	IN, NOT IN
Değer Aralıkları	BETWEEN, NOT BETWEEN
Bilinmeyen Değerler	IS NULL, IS NOT NULL

**ORDER BY İfadesi** : Satırları sıralamak için kullanılır. ASC seçeneği, artan sırada (yani A → Z, 0 → 9), DESC seçeneği azalan sırada (yani Z → A, 9 → 0) sıralama yapmak için kullanılır.

**SELECT** Sütun\_ismi\_1, Sütun\_ismi\_2, .... **FROM** Tablo\_ismi **ORDER BY** Sıralanacak\_sütun [ASC / DESC]

**INNER JOIN ifadesi**

**SELECT** ifadesi, bilgileri yalnızca bir tablodan seçti. Halbuki birçok uygulama birden çok tablo kullandığı için, birden çok tablodan gelen bilgilerin birleştirilmesi gereklidir. 2 çeşit birleştirme vardır. İç ve dış birleştirme. İç birleştirme, yalnızca birleştirme koşulunun **True** (Doğru) olduğu satırları verecektir.

**INSERT ifadesi**

INSERT ifadesi, bir tabloya yeni bir kayıt ilave etmek için kullanılır.

```
INSERT INTO Tablo_ismi (Sütün_ismi_1, Sütün_ismi_2, ..... )  
VALUES (Değer1, Değer2, .....)
```

**UPDATE ifadesi**

UPDATE ifadesi, bir tablodaki bilginin değerlerinin değiştirilmesine olanak sağlar.

```
UPDATE Tablo_ismi  
SET Sütün_ismi_1 = Değer1, Sütün_ismi_2 = Değer2, ..... )  
WHERE Koşul
```

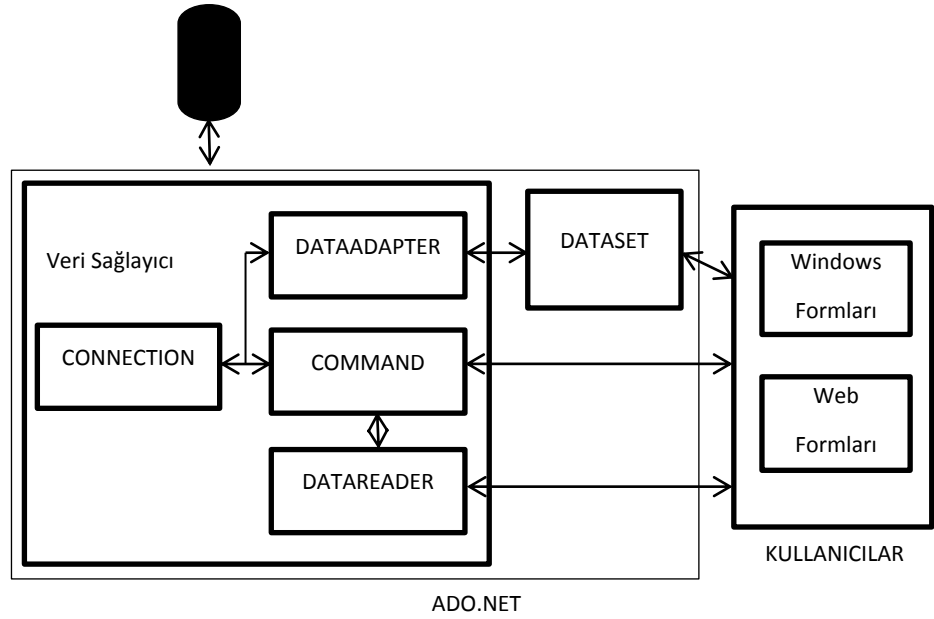
**DELETE ifadesi**

DELETE ifadesi, bir tablodaki satırı silmesini sağlar.

```
DELETE FROM Tablo_ismi  
WHERE Koşul
```

## ADO.NET ve Veritabanı Erişim Araçları

ADO.NET Microsoft'un yeni kuşak veri erişim teknolojisidir. XML desteğine sahiptir. ADO.NET önceki veri erişim metotlarında farklı olarak **Disconnected** (Bağlantısız) veri erişimini kullanır. Bu tür bağlantılarda bağlantı kurulduktan sonra işlem yapılana kadar bağlantı açıktır, işlem bittikten sonra bağlantı kapatılır. Bu da **Connected** (Bağlantılı) veri erişimindeki sistem performansı gibi problemleri yaşamamıza engel olur.



ADO.NET 'te biri (connected) **bağlantılı**, diğeri (disconnected) **bağlantısız** taraf olmak üzere 2 taraf var. Bağlantılı taraf, bir .NET veri sağlayıcısının üzerinden yapılan bağlantıyla çalışan nesneleri içeriyor. **Connection, Command, Data Rader, DataAdapter** bu nesnelerin başlıcalarıdır. Bu nesnelerin ortak özelliği, aktif bir bağlantıya ihtiyaç duymalarıdır.

Diğer tarafta, bağlantısız bir yapı var; **DataSet**. DataSet, veri kaynağı bağlantısından bağımsız olarak tasarlanan bir nesne. Hafızada duran ve istendiğinde XML olarak kalıcı hale getirilebilen ilişkisel bir veritabanı olarak düşünülebilir. Zira DataSet, birden fazla tablolar içerebiliyor; bu tablolar arası ilişkiler, kısıtlamalar, görünümüler tanımlamanıza olanak veriyor.

## VERİ SAĞLAYICILAR

ADO.NET 2 ana veri sağlayıcısına (Data Provider) sahiptir:

- OleDb Veri sağlayıcıları : Access ve diğer veritabanlarına erişmek için kullanılır.
- Sql Veri sağlayıcıları : SQL Server 7.0 ve daha üst versiyon veritabanları için kullanılır.

**Connection** : Veritabanı ile Visual Basic.NET uygulaması arasındaki bağlantıyı sağlar. Connction (bağlantı) bildirimi,

OleDb için → Dim baglanti\_ismi As New OleDbConnection()  
Dim baglanti\_ismi As New OleDbConnection(ConnectionString)

SQL için → Dim baglanti\_ismi As New SqlConnection()  
Dim baglanti\_ismi As New SqlConnection(ConnectionString)

ConnectionString ifadesi ise, tırnak işaretleri arasında birbirinden noktalı virgüllerle ayrılan

#### Parametre = değer

ifadelerinden oluşur. Önemli parametreleri şöyle sıralayabiliriz ;

Provider : Veri sağlayıcının adı

Data Source : Verinin kaynağı (SQL Serverin adı ve adresi)

Initial Catalog : Veritabanının adı

User ID : Kullanıcı adı

Password : Parola

Integrated security veya Trusted Connection : Bağlantının güvenlik tanımı

#### Örnek :

##### OleDb için ;

```
Dim baglanti_ole As New OleDbConnection()
```

```
baglanti_ole.ConnectionString = "Provider = Microsoft.Jet.OLEDB.4.0; Data Source =  
c:\b2002\ogrenci.mdb"
```

```
baglanti_ole.Open()
```

##### SQL için ;

```
Dim baglanti_sql As New SqlConnection("Provider = SQLOLEDB; Data Source = localhost; " &  
"initial catalog = Northwind; User ID = sa, password = aa;")
```

```
baglanti_sql.Open()
```

**Command** : Veritabanına komutları aktarır. Aynı şekilde **SqlCommand** ve **OleDbCommand** şeklinde 2 seçeneği vardır. Genellikle bir önceki bölümde bahsettiğimiz **SQL** komutları kullanılarak erişim yapılacaksa kullanılır.

Bir **Command** nesnesini, aşağıdaki gibi oluşturabiliriz:

```
cmd = new OleDbCommand() veya cmd = new SqlCommand()
```

Oluşturulan bu nesnenin bir **SqlConnection** veya **OleDbConnection** nesnesi ile ilişkilendirilmesi gerekir.

Örnek : `cmd.ExecuteNonQuery()`

**DataReader** : Veritabanındaki bilginin yalnızca görüntüleneceği (düzeltme, silme gibi güncelleştirme işlemlerinin yapılmayacağı) uygulamalarda kullanılır. Böylece **DataSet** kullanılmaz ve bazı kontrol işlemlerine gerek kalmaz. **SqlDataReader** ve **OleDbDataReader** iki çeşidi vardır.

Bir Data **Reader**, bir **Command** nesnesinin **Execute** çağırımı sonucu döndürülür, Bu ADO'daki **Recordset**' in çalışma prensipleriyle benzerlik gösterir ve basit bir şekilde kayıtların üstünde ilerlemeye izin verir.

**Connection** ve **Command** nesnelerinin aksine, **DataReader** nesnesini biz kendimiz oluşturamayız. **DataReader**, **Command** nesnesinin **ExecuteReader()** metodu ile elde edilir. Bu metod, **Command** nesnesinin ifade ettiği sorguyu işletir ve dönen kayıtları temsil edecek **DataReader** nesnesini oluşturur. **DataReader** oluşturulduktan sonra, **Read** metodunu kullanarak, verileri satır olarak elde edebiliriz.

```
Dim Deneme As OleDbDataReader
Deneme = cmd.ExecuteReader()
Do Until Deneme.Read = False
.....
Loop
```



**DataAdapter** : Veritabanından aldığı bilgileri **DataSet** içerisine yerleştirmekte veya **DataSet'** ten aldığı bilgileri geri göndermektedir.

Kullanımında iki adet parametresi vardır. Bunlardan birincisi **DataSet'** e doldurmak istediğimiz veriyi belirten **"SELECT \* from ogrenci"** gibi bir **SQL** cümlesidir. İkincisi ise bağlantı cümlesidir. **DataSet'** e veriyi doldururken de **Fill()** metodu kullanılır.

**Position** : *CurrencyManager* nesnesinin yönettiği listedeki geçerli öğeyi alır ya da ayarlar.

**Count** : *CurrencyManager* nesnesinin yönettiği satırların (kayıtların) sayısı.

**Current** : Veri kaynağındaki geçerli nesnenin değeri.

```
Dim CurMan As CurrencyManager
CurMan = Me.BindingContext(DataSet1, "Ogrenci")
CurMan.Position += 1
```

veya

```
Me.BindingContext(DataSet1, "Ogrenci").Position += 1
```

**DataRelation** : *DataSet*' te yer alan diğer tablo veya tablolarla ilişkiyi düzenler. Tablolar arasında bu ilişkiyi düzenlerken ilk olarak *DataRelation* nesnesi tanımlanır, daha sonra bu ilişki *DataSet* nesnesine ilave 'edilir. Örnek olarak *Kimlik* tablosundaki *Numara* kolunu ile *Notlar* tablosundaki *Numara* kolonunu ilişkilendirelim :

```
Dim DSet As New DataSet()
Dim Bag As DataRelation
Bag = New DataRelation("Numara", DSet.Tables("Kimlik").Columns("Numara"), _
    DSet.Tables("Notlar").Columns("Numara"))
DSet.Relationships.Add(Bag)
```

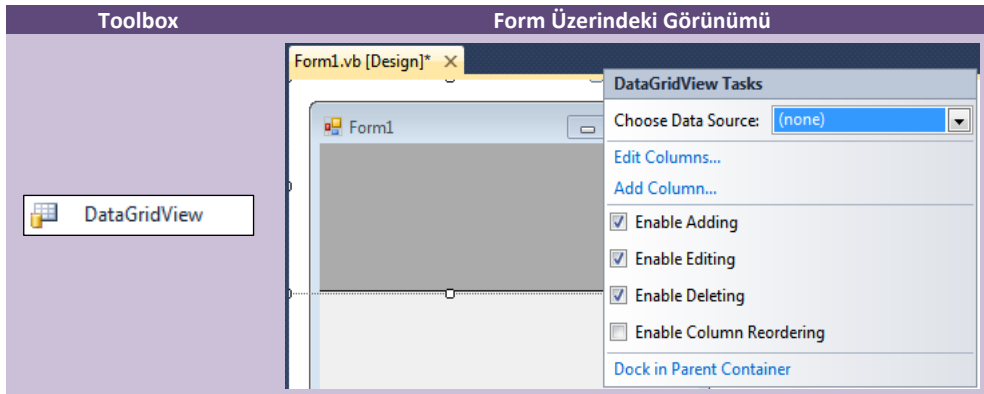
**DataView** : *DataSet* içindeki bilginin istenilen şekilde görüntülenmesini sağlayan nesnedir. Tanımlanması ve kullanımını aşağıdaki gibi basit bir örnekle gösterebiliriz.

```
Dim DV As New DataView (dataset1.Tables("ogrenci"))
DV.RowFilter = " cins = 'k' "
```

Bu satırlardan sonra yalnızca Cinsiyeti Kadın olanlar görüntülenecektir.

## DataGridView Kontrolü

**DataGridView** kontrolü, veriyi Excel benzeri satır ve sütunlardan oluşan bir görünümde sunar. **DataGridView** kontrolü ile bir veya fazla sayıdaki tablolardaki verileri istediğimiz şekilde biçimlendirerek görüntüleyebilir, verilerimizde değişiklik yapabiliriz. Visual Basic 6.0' daki **DataGridView** kontrolüne göre çok farklı özellikler içermektedir.



DataGrid kontrolü, bir Dataset ile ilişkilendirildiğinde, satır ve sütunlar otomatik olarak oluşturulur ve hücreler doldurulur.

#### DataGridView kontrolü ile veritabanı uygulamaları yapmak için:

1. Kontrolün **DataSource** özelliği olarak bağlamak istenilen veriyi içeren **Dataset** nesnesi seçilir.
2. **DataMember** özelliğinde ise bağlamak istenilen tablo seçilir.
3. Veriyi **Dataset'** e doldurmak için, **DataAdapter'**in **Fill** metodu kullanılır.  
**dataAdapter1.Fill (dataset1)**
4. **DataGridView** özelliklerinden yararlanarak Tablo'yu ve kolonları istediğimiz biçimde düzenleyebiliriz.

Tablo ve Kolonların Biçimlendirilmesinde Kullanılan Özellikler	
<b>BackColor</b>	<b>DataGridView'</b> in boş yerlerinin zemin rengi.
<b>BorderStyle</b>	Kenarlık düzenlemesi
<b>ColumnHeadersDefaultCellStyle</b>	<b>DataGridView</b> sütun başlığı için varsayılan stil şekli.
<b>GridColor</b>	<b>Grid</b> çizgilerinin rengi
<b>RowsDefaultCellStyle</b>	<b>DataGridView</b> satır hücreleri için varsayılan stil şekli.
<b>RowHeadersWidth</b>	<b>DataGridView'</b> deki satır başlıklarının genişliği.