6. VERİ ERİŞİM NESNELERİNİN KULLANIMI

6.1. GİRİŞ

Bu bölümde Microsoft Jet veritabanı motoru ve onun programlama modeli olan DAO (Data Access Objects – Veri Erişim Nesneleri) konu edilecektir. Bunun için ilişkisel veritabanı tasarımı, yaratılması, bakımı ve değiştirilmesi konu edilecektir.

Bu bölümde DAO metotları kullanılarak;

- Veritabanı tanımlama
- Alan tanımlama
- İndeks (Dizin) tanımlama
- Tablolar arasında ilişki tanımlama
- Veritabanlarının yapısında değişiklik yapma
- Var olan bir veritabanı üzerinde işlem yapma

konularına açıklık getirilecektir.

Microsoft Jet Veritabanı Motoru

Visual Basic programlama dilindeki veri erişim olanağı, Microsoft Access yazılımının da temel olarak kullandığı Microsoft Jet veritabanı motoruna dayanmaktadır. Jet motoru, verilerin saklanması, geri getirilmesi ve değiştirilmesi için bir mekanizmanın yanısıra güçlü DAO nesne tabanlı arabirimini de kullanıcıya sağlar.

Bir veri tabanı uygulaması üç kesimden oluşur. Bunlar;

- Kullanıcı arabirimi
- Veritabanı motoru
- Verilerin tutulduğu fiziksel ortam kesimleridir.

Veritabanı motoru, yazılım ile fiziksel veritabanı dosyaları arasında iletişimi sağlar. Bu kullanıcıyı veritabanı dosyalarından soyutlar ve hareket serbestliği sağlar. Kullanıcı artık veritabanının türü ile ilgilenmek durumunda değildir. Bütün veritabanı biçimleri için aynı tür erişimler veritabanı motoru tarafından sağlanmaktadır.

Kullanıcı arabirimi, kullanıcının karşı karşıya kaldığı programın dış yüzüdür. Bu kesim, kullanıcının verileri görmesine, değiştirmesine ve veri eklemesine yardımcı olan, formlardan oluşan kesimdir. Bu formların yaptığı işlerle ilgilenen kesim ise

uygulama yazılımının kodudur. Bu kod kullanıcının görsel olarak belirttiği işlemleri veritabanı motoruna iletmekle yükümlüdür.

Jet veritabanı motoru bir takım DLL kütüphaneleri halinde Visual Basic aracılığıyla kullanılır. Jet motoru veritabanı üzerindeki işlemler dışında ayrıca bir sorgu işleyici de barındırır. Bu sorgu işleyici SQL sorgularını veritabanı üzerinde çalıştırır ve sorgu sonuçlarını döndürür.

Veritabanı fiziksel ortamda dosyalar halinde tutulur. Bu dosyalar sadece verileri tutmakla yükümlüdür. Uygulama yazılımları bu dosyaların sadece adı ile muhatap olur. Bu dosyalarda verilerin nasıl tutulduğu veritabanı motorunu ilgilendirir.

Burada bahsedilen üç kesim değişik biçimlerde bölünebilir. Bu üç kesimin bir bilgisayar üzerinde tek bir kullanıcı tarafından kullanılması sözkonusu olabileceği gibi, birbirine ağ yapısı ile bağlı farklı bilgisayarlar üzerinde de bulunmaları mümkündür. Örneğin veritabanı bir ana bilgisayarın üzerinde bulunabilir, kullanıcılar ise bu veritabanına bir ağ üzerinden ulaşabilirler.

Kullanıcının veritabanından uzak (farklı bir bilgisayarda) olduğu durumda iki ayrı yapı sözkonusudur.

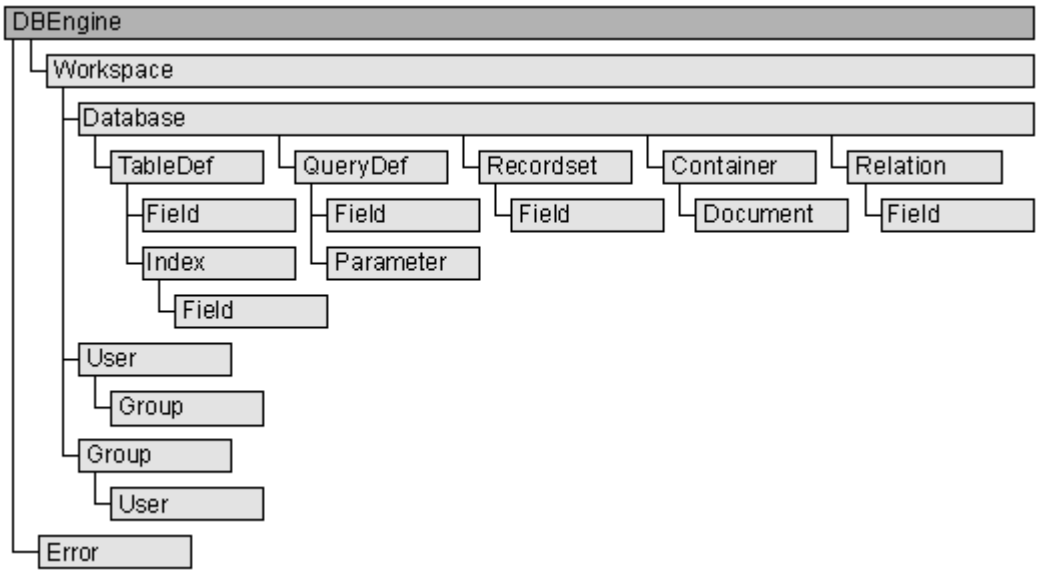
- Uzak veri tabanı sistemi (Remote Database)
- İstemci/Sunucu veritabanı sistemi (Client/Server Database)

Bunlardan birincisinde veritabanı motoru kullanıcı ile aynı bilgisayarda, ikincisinde ise veritabanı motoru veritabanı ile aynı bilgisayar üzerinde bulunur. Veritabanı motoru aynı anda birden fazla kullanıcıdan istek alır ve istedikleri kayıtları veritabanı üzerinde işlem yaparak geri döndürür.

Jet veritabanı motoru Client/Server bir yapıya sahip değildir. Yerel bir veritabanı motorudur. Uygulama ile aynı bilgisayarda bulunur. İşlevleri bir DLL halinde bulunur. Eğer bir uygulama programının farklı bilgisayarlar üzerinde kopyaları varsa herbirinin kendi Jet veritabanı motoru DLL dosyalarına sahip olması gerekir.

Visual Basic kullanarak Client/Server veritabanları ile çalışmak da mümkündür. Bunun için ODBC standardı kullanılarak sorgular doğrudan ODBC server olarak adlandırılan veritabanı sunucusuna gönderilebilir.

Veri erişim nesnesi modeli Jet veritabanı motorunun veritabanı motorudur. Bu model aşağıdaki çizimdeki sıradüzensel yapı ile ifade edilmektedir.



Bu yapıdaki elemanların herbiri aslında bir sınıfı temsil etmektedir. Bu sınıfın bir nesnesini yaratmak için ise, örneğin;

Dim MyWs as Workspace

gibi bir komut kullanmak gerekir. Yukarıdaki şemada DBEngine dışındaki elemanlar hem küme(Collection) hem de nesne olarak kullanılabilirler. Kümelerin elemanlarına sıfırdan başlayan bir dizinli yapı ile erişmek mümkündür. Bu erişim için aşağıdaki gösterim bir örnek olarak verilebilir.

DBEngine.Workspaces(0).Databases(0).TableDefs(0).Fields("MüşteriNo")

Şu ana kadar veritabanı kavramlarına genel bir giriş yapılmış oldu. Veritabanları üzerinde işlem yapılırken iki tür dilden bahsedilir. Bunlar;

- DDL (Data Definition Language – Veri tanımlama dili)
- DML (Data Manipulation Language – Veri işleme dili)

Bu sözkonusu olan farklı iki dil olduğu anlamına gelmez. Bu sadece verilerle işlem yapılırken kullanılırken yapılan bir gruplamadır.

Veri Tanımlama(DDL): bir veritabanının tanımlaması ve yaratılması için gerekli özellikleri ve metotları içerir. Veritabanının yaratılması bir defaya mahsus yapılan bir işlemdir. Bir defa veritabanı yaratıldıktan sonra, onu açmak tüm yapısına erişmek anlamına gelir.

Veri İşleme(DML): varolan veritabanlarına erişebilen ve onların üzerinde işlem yapabilen uygulama yazılımları yazmak için gerekli özellikleri ve metotları içerir. Bu

veritabanını sorgulama, tabloları üzerinde dolaşma, tutanaklar üzerinde değişiklik, ekleme, silme gibi işlemler yapmayı içerir.

Var olan veritabanlarını kullanmak için sadece DML yeterlidir. Fakat DDL metotlarını ve özelliklerini anlamak veritabanının yapısını daha iyi kavrama ve veritabanı üzerinde daha rahat işlem yapabilme olanağı sağlar.

6.2. RECORDSET YAPISI İLE ÇALIŞMAK

Recordset yapısı veritabanına erişimi sağlayan bir nesne yapısıdır. Bir Recordset nesnesi, bir tablodaki tutanakları veya bir sorgunun sonucunda döndürülen tutanak kümelerini temsil eder. Beş farklı Recordset nesnesi vardır. Bunların herbiri farklı özellikler gösterir. Bunlar, **Table** , **Dynaset** , **Snapshot** , **Dynamic** ve **Forward-only** nesneleridir.

Table (Tablo)

Kullanılan veritabanı üzerindeki bir tabloyu belirtir. Bu türden bir Recordset yaratıldığında, veritabanı motoru veritabanındaki bir tabloyu açar ve işlemler onun üzerinde gerçekleştirilir. Bir tablo türündeki Recordset ilişkilendirildiği veritabanı tablosunun indeks yapısını kullanabilir. Bu durum, hızlı arama yapabilme olanağı sağlar.

Dynaset

Kullanılan veritabanı üzerinde yerel olarak var olan veya bu veritabanına bağlı olan tablolar ile bir sorgu sonucunda oluşturulan tabloları da temsil edebilir. Genelde bir veya birden fazla tablonun tutanaklarına referans kümeleri içerir. Bu yapı veritabanı üzerinde çok esnek bir erişim sağlar, farklı türden veritabanları üzerinde aynı Recordset yapısı kullanarak işlem yapmaya olanak sağlar. Bu avantajının yanısıra birden fazla tabloya çok sayıda bağ içerdiğinden dolayı çalışma esnasında yavaş olduğu görülür.

Snapshot

Yaratıldığı anda ilişkilendirildiği verilerin sabit bir kopyasını barındırır. Jet veritabanı motoru tarafından kullanılan snapshot yapısında değişiklik yapılamaz. Ancak ODBC veritabanları sunucunun olanaklarına göre snapshot yapısı kullanarak değişiklik yapmaya izin verebilir. Snapshot işlem miktarını azalttığı için daha hızlı bir yapıdır. Bir sorgu sonucu bu yapı ile çok hızlı bir şekilde elde edilebilir.

Forward-only

Forward-scrolling snapshot veya forward-only snapshot olarak adlandırıldığı da olur. Snapshot yapısının sağladığı olanakların bir alt kümesini sunar. En az işlevselliği olan Recordset yapısıdır, bu yüzden en hızlı işlem yapan yapıdır. Bu yapıda da snapshot yapısında olduğu gibi değiştirme yapılamaz. Bunun yanısıra bir kısıtlama daha vardır, bu da sadece ileriye doğru hareket edebilme özelliğidir.

Dynamic

Bir veya birden fazla tabloyu içeren bir sorgu sonucunu tutmak için kullanılır. Bu yapıda ekleme, silme değiştirme gibi özellikler de sağlanır. Kullanıldığı sırada tablolardaki değişiklik de hemen bu yapıya yansır. Bu yapı ODBC veritabanılarındaki Dynamic Cursor yapısını temsil eder.

Recordset türleri kullanılırken dikkat edilmesi gereken bir unsur, Snapshot yapısı kullanılırken verilerin tümünün bir kopyasının alındığıdır. Bu haliyle bazen bir Dynaset yapısı, Snapshot yapısından daha hızlı olabilir.

Genel olarak eğer Table türü bir Recordset kullanmak mümkünse öncelikli olarak bu yapı tercih edilmelidir.

Veritabanına erişim için Recordset yapısının kullanımını örneklemeden önce veritabanının nasıl tasarlandığını ve yaratıldığını bir örnekle açıklayalım.

6.2.1 VERİTABANI TASARIMI VE YARATILMASI

Veritabanı tasarlama ve yaratma konusu işlenirken bir veritabanının tasarlanması şarttır. Bunun için aşağıdaki kısıtlı olarak tasarlanmış PERSONEL veritabanını kullanalım. Bu veritabanı bir kurumda çalışan personel ile ilgili birtakım bilgileri saklamak için kullanılıyor varsayalım.

PERSONEL veritabanı

Bu veritabanında aşağıdaki tabloların bulunduğunu düşünelim.

1. PERSONEL_BIL(SICNO, AD_SOYAD, DTAR, DYER)

SICNO: Sicil numarası, anahtar

AD_SOYAD: Personelin adı ve soyadı

DTAR: Doğum tarihi

DYER: Doğum yeri

2. PERSONEL_GOREV(SICNO, CYIL, GNO, DERECE, MNO)

SICNO: Sicil numarası, anahtar

CYIL: Kurumda çalıştığı yıl sayısı

GNO: Görev numarası

DERECE: Personelin kadro derecesi

MNO: Meslek numarası

3. MESLEKLER(MNO, MADI, ACIKLAMA)

MNO: Meslek numarası, anahtar

MADI: Meslek adı

ACIKLAMA: Meslekle ilgili açıklama

4. GOREVLER(GNO, GADI)

GNO: Görev numarası

GADI: Görev numarası

Bu veritabanını yaratmak için Standart EXE türünde bir proje yaratın. Üzerine iki tane CommandButton yerleştirin. Caption özelliklerini sırasıyla **Yarat** ve **CIKIŞ** olarak değiştirin ve aşağıdaki kodu forma ekleyin.

```
Option Explicit
Private Sub Command1_Click()
Dim ws As Workspace
Dim db As Database
Dim tdf As TableDef
Dim fld As Field
Set ws = DBEngine.Workspaces(0)
Set db = ws.CreateDatabase(App.Path & _
"\PERSONEL.MDB", dbLangTurkish)
'PERSONEL_BIL tablosu
Set tdf = db.CreateTableDef("personel_bil")
Set fld = tdf.CreateField("sicno", dbText, 10)
tdf.Fields.Append fld
Set fld = tdf.CreateField("ad_soyad", dbText, 30)
tdf.Fields.Append fld
Set fld = tdf.CreateField("dtar", dbDate)
tdf.Fields.Append fld
Set fld = tdf.CreateField("dyer", dbText, 15)
```

```
tdf.Fields.Append fld
db.TableDefs.Append tdf
'PERSONEL_GOREV tablosu
Set tdf = db.CreateTableDef("personel_gorev")
Set fld = tdf.CreateField("sicno", dbText, 10)
tdf.Fields.Append fld
Set fld = tdf.CreateField("CYIL", dbInteger)
tdf.Fields.Append fld
Set fld = tdf.CreateField("gno", dbInteger)
tdf.Fields.Append fld
Set fld = tdf.CreateField("derece", dbInteger)
tdf.Fields.Append fld
Set fld = tdf.CreateField("mno", dbInteger)
tdf.Fields.Append fld
db.TableDefs.Append tdf
'MESLEKLER tablosu
Set tdf = db.CreateTableDef("gorevler")
Set fld = tdf.CreateField("mno", dbInteger)
tdf.Fields.Append fld
Set fld = tdf.CreateField("MADI", dbText, 40)
tdf.Fields.Append fld
db.TableDefs.Append tdf
'GOREVLER tablosu
Set tdf = db.CreateTableDef("meslekler")
Set fld = tdf.CreateField("gno", dbInteger)
tdf.Fields.Append fld
Set fld = tdf.CreateField("GADI", dbText, 40)
```

```
tdf.Fields.Append fld
db.TableDefs.Append tdf
db.Close
Label1.Caption = "PERSONEL VERİTABANI YARATILDI"
Command1.Enabled = False
End Sub
Private Sub Command2_Click()
End
End Sub
```

Program çalıştırıldıktan sonra aşağıdaki form görüntüsü ekrana çıkar, fakat üstteki yazı yoktur. Yarat düğmesine basıldıktan sonra üstteki yazı belirir. Daha sonra yapılması gerken ÇIKIŞ düğmesine basarak programdan çıkmaktır.

Programdan çıktıktan sonra programla aynı dizinde PERSONEL.MDB adlı bir veritabanı kütüğü yaratılmış olur. Bu veritabanının alanlarını görmek için Visual Basic dizinindeki Visdata.exe programı kullanılabilir.

İndeks Yaratma.

Veritabanını yarattıktan sonra sıra indeks ekleme işlemine geldi.

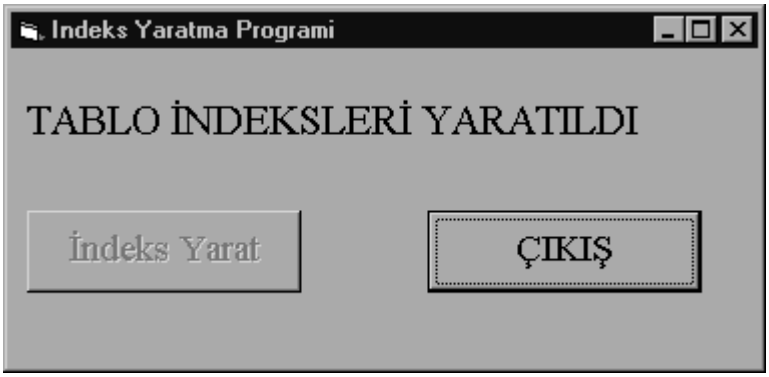
İndeksler veritabanı üzerinde arama yapılırken hızlı ve kolay erişim sağlayan veritabanı öğeleridir.

Yukarıdaki projenin Command1_Click olay yordamını kaldırın ve form üzerindeki **Yarat** butonunun **Caption** özelliğini **İndex Yarat** olarak değiştirin ve aşağıdaki kodu Command1_Click olayına ekleyin.

```
Private Sub Command1_Click()
Dim ws As Workspace
Dim db As Database
Dim tdf As TableDef
Dim fld As Field
Dim idx As Index
Set ws = DBEngine.Workspaces(0)
Set db = ws.OpenDatabase(App.Path & "\PERSONEL.MDB")
'PERSONEL_BIL tablosu için indeks yaratma
Set tdf = db.TableDefs("personel_bil")
Set idx = tdf.CreateIndex("ind_sicno")
Set fld = idx.CreateField("sicno")
idx.Fields.Append fld
Set fld = Nothing
idx.Primary = True
idx.Unique = True
tdf.Indexes.Append idx
'PERSONLE_GOREV tablosu için indeks yaratma
Set tdf = db.TableDefs("personel_gorev")
Set idx = tdf.CreateIndex("ind_sicno")
Set fld = idx.CreateField("sicno")
idx.Fields.Append fld
Set fld = Nothing
idx.Primary = True
idx.Unique = True
tdf.Indexes.Append idx
```

```
'MESLEKLER tablosu için indeks yaratma
Set tdf = db.TableDefs("meslekler")
Set idx = tdf.CreateIndex("ind_mno")
Set fld = idx.CreateField("mno")
idx.Fields.Append fld
Set fld = Nothing
idx.Primary = True
idx.Unique = True
tdf.Indexes.Append idx
'GOREVLER tablosu için indeks yaratma
Set tdf = db.TableDefs("gorevler")
Set idx = tdf.CreateIndex("ind_gno")
Set fld = idx.CreateField("gno")
idx.Fields.Append fld
Set fld = Nothing
idx.Primary = True
idx.Unique = True
tdf.Indexes.Append idx
db.Close
Label1.Caption = "TABLO İNDEKSLERİ YARATILDI"
Command1.Enabled = False
End Sub
```

Program çalıştırıldıktan sonra aşağıdaki form görüntüsü ekrana çıkar, fakat üstteki yazı yoktur. İndeks yarat düğmesine basıldıktan sonra üstteki yazı belirir. Daha sonra yapılması gereken ÇIKIŞ düğmesine basarak programdan çıkmaktır.



Programdan çıktıktan sonra Visual Basic dizinindeki Visdata.exe programı kullanılarak indekslerin yaratılıp yaratılmadığı kontrol edilebilir.

İlişki Yaratma

İki farklı tabloda aynı olan alanların tutarlılığını denetlemek için bu tablolar arasında bağ kurma işlemine ilişki yaratma denir.

Yukarıdaki veritabanı için konuşacak olursak, PERSONEL_BIL tablosunda olmayan bir personel, PERSONEL_GOREV tablosunda da olmamalıdır. Bunu sağlamak için iki tablo arasında bir ilişki tanımlamak şarttır.

Jet veritabanı motoru ilişki tanımlamak için Relation diye bir nesne sağlar.

Bir veritabanına ilişki eklemek için:

1- Database nesnesinin CreateRelation metodunu kullanarak bir ilişki nesnesi yaratın.

2- İlişkinin dayandığı alanları belirlemek için Relation nesnesinin CreateField metodunu kullanarak bir alan yaratın.

3- Yaratılan bu alanı yaratılmış 1. Adımda yaratılan ilişki nesnesine ekleyin. Sonra bu ilişkiyi veritabanına ekleyin.

Bu adımların uygulanmış hali olan aşağıdaki kodu, yukarıdaki formun Command1_Click olay yordamını silerek yerine yazın.

```
Private Sub Command1_Click()
Dim ws As Workspace
```

```
Dim db As Database
Dim fld As Field
Dim rel As Relation
Set ws = DBEngine.Workspaces(0)
Set db = ws.OpenDatabase(App.Path & "\PERSONEL.MDB")
'PERSONEL_BIL ile PERSONEL_GOREV tablosu arasında
'p_gorevi (personelin görevi) ilişkisi yaratma
Set tdf = db.CreateRelation("p_gorevi")
rel.Table = "personel_bil"
rel.ForeignTable = "personel_gorev"
Set fld = rel.CreateField("sicno")
fld.ForeignName = "sicno"
rel.Fields.Append fld
db.Relations.Append rel
'PERSONEL_GOREV ile GOREVLER tablosu arasında
'g_ad (görev adı) ilişkisi yaratma
Set tdf = db.CreateRelation("g_ad")
rel.Table = "gorevler"
rel.ForeignTable = "personel_gorev"
Set fld = rel.CreateField("gno")
fld.ForeignName = "gno"
rel.Fields.Append fld
db.Relations.Append rel
```

```
'PERSONEL_GOREV ile MESLEKLER tablosu arasında
'm_ad (meslek adı) ilişkisi yaratma
Set tdf = db.CreateRelation("m_ad")
rel.Table = "meslekler"
rel.ForeignTable = "personel_gorev"
Set fld = rel.CreateField("mno")
fld.ForeignName = "mno"
rel.Fields.Append fld
db.Relations.Append rel
db.Close
Label1.Caption = "İLİŞKİLER YARATILDI"
Command1.Enabled = False
End Sub
```

Program yine yukarıdakilere benzer bir biçimde çalışacak ve belirtilen üç ilişkiyi yaratacaktır. Bu ilişkilerin veritabanına eklendiği yine VISDATA programı kullanılarak test edilebilir. Bu ilişkiler şunlardır.

1- PERSONEL_GOREV ile GOREVLER tablosu arasında g_ad (görev adı) ilişkisi.

2- PERSONEL_GOREV ile GOREVLER tablosu arasında 'g_ad (görev adı) ilişkisi.

3- PERSONEL_GOREV ile MESLEKLER tablosu arasında m_ad (meslek adı)

ilişkisi.

VISDATA ile bakıldığında bu ilişkilerin PERSONEL_GOREV tablosuna birer indeks olarak eklendiği görülecektir.

Bu ilişkiler veritabanının tutarlılığını devam ettirmesine yardımcı olur. Örneğin bir personelin GOREVLER tablosunda olmayan bir görevi olamaz.

6.2.2. VERİTABANINA ERİŞİM

Veritabanı üzerinde DataControl nesnesi kullanmadan da dolaşmak mümkündür. Bunun için veritabanı ile kontrol nesneleri arasındaki ilişki elle kurulmalıdır. Bunu örneklemek için aşağıdaki proje verilebilir.

1- Formun üstüne aşağıdaki nesneleri ileride verilecek olan form nesnesini referans alarak yerleştirin.

2-

Form	Name	frmDolas
	Caption	Veritabanini Dolasma
CommandButton	Name	Command1 (Control dizisi, 4 elemanlı, Caption özellikleri: |<, <, >, >|
TextBox	Name	Text1 (Control dizisi, 4 elemanlı)
CommandButton	Name	cmdKaydet
	Caption	&Kaydet

3- Projeye bir Class Module ekleyin ve clsDolas.bas olarak kaydedin. Daha sonra aşağıdaki kodu bu Class Module içine ekleyin.

```
Private db As Database
Private rs As Recordset
Private degisti As Boolean
Private mmad As String
Private myil As String
Private mISBN As String
Private mbno As String
Public Enum hareket
ilk = 1
```

```
 son = 2
birsonraki = 3
bironceki = 4
End Enum
Public Enum hata
yhareket = vbObjectError + 1000 + 11
tutanakyok = vbObjectError + 1000 + 12
End Enum
Private Sub class_Initialize()
Dim dosyaAdi As String
dosyaAdi = "c:\devstudio\vb\biblio.mdb"
Set db = DBEngine.Workspaces(0).OpenDatabase(dosyaAdi)
Set rs = db.OpenRecordset("Title", dbOpenDynaset, _
dbSeeChanges, dbOptimistic)
If rs.BOF Then HataVer yhareket
TutanakOku
End Sub
Private Sub Class_Terminate()
rs.Close
Set rs = Nothing
db.Close
Set db = Nothing
End Sub
Private Sub HataVer(hatano As hata)
Dim tanim As String
Dim aciklama As String
Select Case hatano
Case yhareket: tanim = "YANLIS HAREKET"
```

```
Case tutanakyok: tanim = "TUTANAK kitaplar tablosunda yok"
Case Else: tanim = "TANIMSIZ HATA"
End Select
aciklama = App.EXEName & ".clsDolas"
End Sub
Private Sub TutanakOku()
mISBN = rs![ISBN] & " "
mad = rs![Title] & " "
myil = rs![Year Published] & " "
mbno = rs![PubID] & " "
End Sub
Private Sub TutanakGunle()
On Error GoTo pHata
rs.Edit
rs![ISBN] = mISBN
rs![Title] = mad
rs![Year Published] = myil
rs![PubID] = mbno
rs.Update
degisti = False
Exit Sub
pHata:
rs.MovePrevious
rs.MoveNext
Err.Raise Err.Number, Err.Source, Err.Description, _
Err.HelpFile, Err.HelpContext
End Sub
Public Property Get ad() As String
```

```
ad = mad
End Property
Public Property Let ad(sad As String)
mad = sad
degisti = True
End Property
Public Property Get yil() As String
yil = myil
End Property
Public Property Let yil(syil As String)
myil = syil
degisti = True
End Property
Public Property Get ISBN() As String
ISBN = mISBN
End Property
Public Property Let ISBN(sISBN As String)
mISBN = sISBN
degisti = True
End Property
Public Property Get bno() As String
bno = mbno
End Property
Public Property Let bno(sbno As String)
mbno = sbno
degisti = True
EndProperty
```

```
Public Property Get degistimi() As Boolean
degistimi = degisti
End Property
Public Sub git(htur As hareket)
On Error GoTo phata:
Select Case htur
Case ilk: rs.MoveFirst
Case son: rs.MoveLast
Case birsonraki: rs.MoveNext
Case bironceki: rs.MovePrevious
Case Else: HataVer yhareket
End Select
TutanakOku
phata: If rs.EOF Or rs.BOF Then HataVer yhareket
End Sub
Public Sub kaydet()
If degisti Then TutanakGunle
End Sub
```

Bu kodu biraz inceleyecek olursak, Class_Initialize olayında veritabanı açılıyor, Class_Terminate olayında ise veritabanı kapatılıyor. Daha sonra tutanak okuma, günleme yordamları yer almaktadır. Burada kullanılan veritabanı BIBLIO.MDB veritabanıdır. Recordset olarak da bu veritabanının Titles (Kitaplar) tablosu kullanılmaktadır.

Kodun geri kalan kesiminde Recordset yapısının her bir alanı için bir Property tanımlandığı görülebilir. Property Let yordamlarında aktif tutanakta değişiklik yapıldığını belirten degisti adlı bir değişkene True değeri aktarılmaktadır.

degistimi adlı Property aktif tutanakta değişiklik yapılıp yapılmadığı bilgisini döndüren sadece okunabilir bir özeliktir.

Yukarıdaki kodda görülebileceği gibi veritabanına erişimi destekleyen yordamların yanısıra hata durumlarını ele alan ve hatalı bir duruma girildiğinde hata veren yordamlar da bulunmaktadır. Bu yordamlar kendi hata mesajlarımızı yazmak ve hatalı durumalara gelindiğinde bu mesajları kullanmak amacıyla yazılmıştır.

Bu erişim sınıfını tanımladıktan sonra, şimdi de frmDolas formunda yapılması gereken işlemlere dönelim. Aşağıdaki kodu frmDolas formuna ekleyin.

```
Private Kitaplar As clsDolas
Private okunuyor As Boolean
Private Sub Form_Load()
On Error GoTo phata
Set Kitaplar = New clsDolas
VeriAl
pcik: Exit Sub
phata:
MsgBox Err.Description, vbExclamation
Unload Me
Resume pcik
End Sub
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
On Error GoTo phata
Kitaplar.kaydet
pcik: Exit Sub
phata:
If MsgBox("Olusan hata:" & vbCrLf & Err.Description & vbCrLf & _
"devam ederseniz, yaptiginiz degisiklikler kaybolur" & vbCrLf & _
```

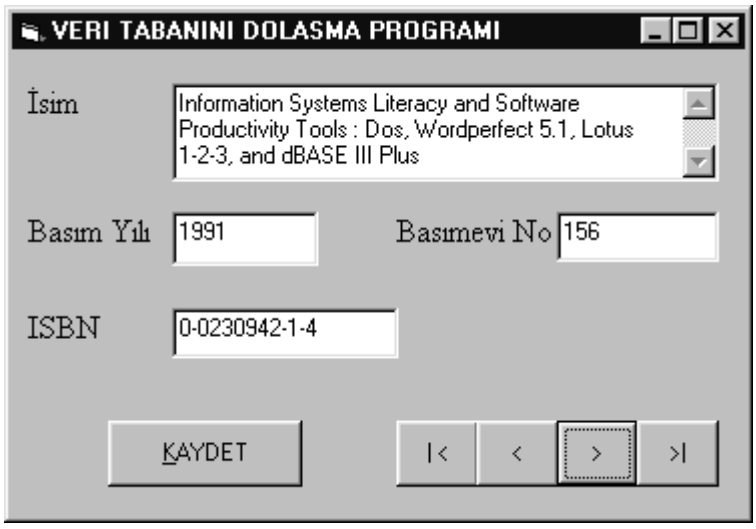
```
"Devam etmek istiyor musunuz?", vbQuestion Or vbYesNo Or _
vbDefaultButton2) = vbNo Then Cancel = True
Resume pcik
End Sub

Private Sub cmdKaydet_Click()
On Error GoTo phata
Kitaplar.kaydet
VeriAl
pcik: Exit Sub
phata:
 MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub Command1_Click(Index As Integer)
On Error GoTo phata
Kitaplar.kaydet
Select Case Index
Case 0: Kitaplar.git ilk
Case 1: Kitaplar.git bironceki
Case 2: Kitaplar.git birsonraki
Case 3: Kitaplar.git son
End Select
pcik: Exit Sub
phata:
```

```
MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub Text1_Change(Index As Integer)
On Error GoTo phata
If Not okunuyor Then
Select Case Index
Case 0: Kitaplar.ad = Text1(Index).Text
Case 1: Kitaplar.yil = Text1(Index).Text
Case 2: Kitaplar.bno = Text1(Index).Text
Case 3: Kitaplar.ISBN = Text1(Index).Text
End Select
End If
pcik: Exit Sub
phata:
MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub VeriAl()
okunuyor = True
Text1(0).Text = Kitaplar.ad
Text1(1).Text = Kitaplar.yil
Text1(2).Text = Kitaplar.bno
Text1(3).Text = Kitaplar.ISBN
okunuyor = False
End Sub
```

Burada görüleceği gibi veritabanı üzerinde doğrudan hiç bir işlem yapılmamaktadır. Veritabanı erişimleri formun kodu içinde yapmaktansa sınıfın içinde yapıp bunu formun kodundan saklamak kod yazımında önemli bir kolaylık ve esneklik sağlamaktadır. Bu nesneye yönelik yaklaşımın sağladığı avantajlara bir örnek olarak verilebilir. Formun kodunda yapılan işlemleri sırayla açıklayacak olursak;

- Form yüklenirken clsDolas sınıfından bir nesne yaratılmakta ve içine veriler okunmaktadır.
- Daha sonra form bellekten silinirken değişiklikler kaydedilmekte ve hata durumunda ne yapılacağı bilgisi kullanıcıya bırakılmaktadır.
- Kaydet butonu ile değişiklikler kaydedilmektedir.
- Hareket tuşlarına basıldığında yapılması gerekenler yapılmaktadır.
- Metin kutuları üzerinde yapılan değişiklikler doğrudan kayıt nesnesinin özelliklerine aktarılır.
- VeriAl yordamında ise kayıt nesnesinin özellikleri metin kutularına aktarılmaktadır.

Programın çalışması aşağıdaki gibi olacaktır.



6.2.3. VERİTABANI ÜZERİNDE EKLEME VE SİLME İŞLEMLERİ

Yukarıda hazır bir veritabanı olan BIBLIO.MDB üzerinde tanımladığımız sınıfı kendi veritabanımız olan PERSONEL veritabanı üzerinde yeniden tanımlayalım. Bu sınıfı personel_bil tablosu üzerinde yeniden tanımlamak için kod aşağıdaki gibi değiştirilmelidir.

```
Private db As Database
Private rs As Recordset
Private degisti As Boolean
Private myeni As Boolean

Private msicno As String
Private mad_soyad As String
Private mdyer As String
Private mdtar As String

Public Enum hareket
ilk = 1
son = 2
birsonraki = 3
bironceki = 4
End Enum
Public Enum hata
yhareket = vbObjectError + 1000 + 11
tutanakyok = vbObjectError + 1000 + 12
End Enum
Private Sub class_Initialize()
Dim dosyaAdi As String
dosyaAdi = App.Path & "\personel.mdb"
Set db = DBEngine.Workspaces(0).OpenDatabase(dosyaAdi)
Set rs = db.OpenRecordset("personel_bil", dbOpenDynaset, _

dbSeeChanges, dbOptimistic)
```

```
If rs.BOF And rs.EOF Then
Yeni
Else
TutanakOku
End If
End Sub
Private Sub Class_Terminate()
rs.Close
Set rs = Nothing
db.Close
Set db = Nothing
End Sub
Private Sub HataVer(hatano As hata)
Dim tanim As String
Dim aciklama As String
Select Case hatano
Case yhareket: tanim = "YANLIS HAREKET"
Case tutanakyok: tanim = "tutanak personel_bil tablosunda yok"
Case Else: tanim = "TANIMSIZ HATA"
End Select
aciklama = App.EXEName & ".clsPBil"
End Sub
Private Sub TutanakOku()
mdyer = rs![dyer] & " "
msicno = rs![sicno] & " "
```

```
mad_soyad = rs![ad_soyad] & " "
mdtar = rs![dtar] & " "
MsgBox aciklama & vbCrLf & tanim
End Sub
Private Sub TutanakGunle()
On Error GoTo phata
rs.Edit
rs![dyer] = mdyer
rs![sicno] = msicno
rs![ad_soyad] = mad_soyad
rs![dtar] = mdtar
rs.Update
degisti = False
Exit Sub
phata:
rs.MovePrevious
rs.MoveNext
Err.Raise Err.Number, Err.Source, Err.Description, _
Err.HelpFile, Err.HelpContext
End Sub

Public Property Get sicno() As String
sicno = msicno
End Property
Public Property Let sicno(ssicno As String)
msicno = ssicno
degisti = True
End Property
```

```
Public Property Get ad_soyad() As String
ad_soyad = mad_soyad
End Property
Public Property Let ad_soyad(sad_soyad As String)
mad_soyad = sad_soyad
degisti = True
End Property
Public Property Get dyer() As String
dyer = mdyer
End Property
Public Property Let dyer(sdyer As String)
mdyer = sdyer
degisti = True
End Property
Public Property Get dtar() As String
dtar = mdtar
End Property

Public Property Let dtar(sdtar As String)
mdtar = sdtar
degisti = True
End Property
Public Property Get degistimi() As Boolean
degistimi = degisti
End Property
```

```
Public Sub git(htur As hareket)
On Error GoTo phata:
Select Case htur
Case ilk: rs.MoveFirst
Case son: rs.MoveLast
Case birsonraki: rs.MoveNext
Case bironceki: rs.MovePrevious
Case Else: HataVer yhareket
End Select
TutanakOku
phata:
If rs.EOF Or rs.BOF Then HataVer yhareket
End Sub
Public Sub kaydet()
If degisti Then
If myeni Then
YeniTutanak
Else: TutanakGunle
End If
End If
End Sub
Private Sub YeniTutanak()
rs.AddNew
rs![dyer] = mdyer
rs![sicno] = msicno
rs![ad_soyad] = mad_soyad
```

```
rs![dtar] = mdtar
rs.Update
rs.Bookmark = rs.LastModified
myeni = False
degisti = False
End Sub
Public Property Get yenimi() As Boolean
yenimi = myeni
End Property
Public Sub Yeni()
mdyer = " "
msicno = " "
mad_soyad = " "
mdtar = " "
myeni = True
End Sub
Public Sub sil()
rs.Delete
degisti = False
myeni = False
rs.MovePrevious
If rs.BOF Then
If Not rs.EOF Then
rs.MoveFirst
Else: HataVer tutanakyok
End If
```

```
End If
TutanakOku
End Sub
```

Bu kod hemen hemen daha önceki sınıf tanımıyla aynıdır. Yapılan değişiklik tablo alanlarına göre değişkenlerin ve yordamların yeniden adlandırılmasıdır. Bunun yanısıra sınıfın içine iki yeni metot daha eklenmiştir. Bunlar Yeni tutanak ekleme ve tablodan tutanak silme yordamlarıdır. Ayrıca Class_Initialize yordamında eğer tablo boşsa yeni tutanak ekleme yordamı çağrılmaktadır.

Bu sınıftan yaratılmış bir nesne kullanarak tabloyu dolaşmayı ve tablo üzerinde ekleme, silme işlemlerini yapan formun tasarımı daha önceki örnekte kullanılan form üzerinde aşağıdaki değişiklikleri yaparak mümkündür.

Form	Name	frmPBil
	Caption	Veritabanını İşleme
CommandButton	Name	cmdKaydet
CommandButton	Name	cmdEkle
CommandButton	Name	cmdSil

Formun tasarımını yapmak için ileride verilecek olan form tasarımı referans alınabilir.

Forma eklenecek kod aşağıdaki gibi olabilir.

```
Private PBil As clsPBil
Private okunuyor As Boolean
Private Sub cmdEkle_Click()
On Error GoTo phata
PBil.kaydet
PBil.Yeni
VeriAl
pcik: Exit Sub
```

```
phata:
MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub cmdSil_Click()
On Error GoTo phata
PBil.kaydet
PBil.sil
VeriAl
pcik: Exit Sub
phata:
Select Case Err.Number
Case tutanakyok
PBil.Yeni
Resume Next
Case Else
MsgBox Err.Description, vbExclamation
Resume pcik
End Select
End Sub
Private Sub Form_Load()
On Error GoTo phata
Set PBil = New clsPBil
VeriAl
pcik: Exit Sub
phata:
Select Case Err.Number
```

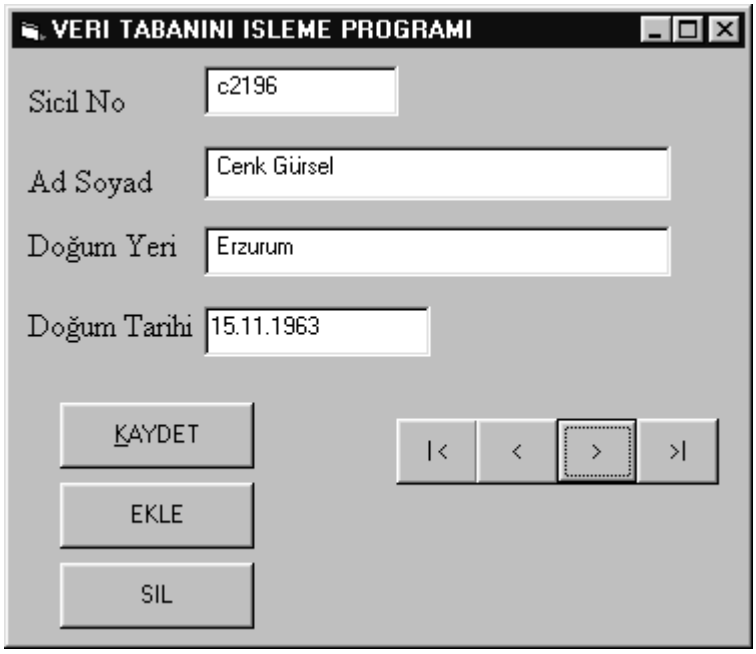
```
Case tutanakyok
PBil.Yeni
Resume Next
Case Else
MsgBox Err.Description, vbExclamation
Unload Me
Resume pcik
End Select
End Sub
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
On Error GoTo phata
PBil.kaydet
pcik: Exit Sub
phata:
If MsgBox("Olusan hata:" & vbCrLf & Err.Description & vbCrLf & _
"devam ederseniz, yaptiginiz degisiklikler kaybolur" & vbCrLf & _
"Çikmak istiyor musunuz?", vbQuestion Or vbYesNo Or _
vbDefaultButton2) = vbNo Then Cancel = True
Resume pcik
End Sub
Private Sub cmdKaydet_Click()
On Error GoTo phata
PBil.kaydet
VeriAl
```

```
pcik: Exit Sub
phata:
MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub Command1_Click(Index As Integer)
On Error GoTo phata
PBil.kaydet
Select Case Index
Case 0: PBil.git ilk
Case 1: PBil.git bironceki
Case 2: PBil.git birsonraki
Case 3: PBil.git son
End Select
VeriAl
pcik: Exit Sub
phata:
MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub Text1_Change(Index As Integer)
On Error GoTo phata
If Not okunuyor Then
Select Case Index
Case 0: PBil.sicno = Text1(Index).Text
Case 1: PBil.ad_soyad = Text1(Index).Text
```

```
Case 2: PBil.dyer = Text1(Index).Text
Case 3: PBil.dtar = Text1(Index).Text
End Select
End If
pcik: Exit Sub
phata:
MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub VeriAl()
okunuyor = True
Text1(0).Text = PBil.sicno
Text1(1).Text = PBil.ad_soyad
Text1(2).Text = PBil.dyer
Text1(3).Text = PBil.dtar
okunuyor = False
End Sub
```

Bu form için yapılanlar da sınıf için yapılanların aynıdır. Yeni eklenen kısımlar Ekleme ve Silme butonlarına basıldığında ilgili clsPBil sınıfının metotlarının çağırılmasıdır.

Formun çalışır haldeki görüntüsü aşağıdaki gibidir.



6.2.4. VERİTABANI ÜZERİNDE ARAMA İŞLEMLERİ

Arama işlemi kullanmak için bir önceki örnekte kullandiğımız clsPBil sınıfını olduğu gibi kullanabiliriz. Gerekli olan frmPBil formuna bir buton daha eklemek ve bu butona işlev kazandırmaktır. Bunu sağlamak için aşağıdaki kodu forma ekleyin.

```
Private Sub cmdBul_Click()
Dim ad As String
Dim bulundu As Boolean
On Error GoTo phata
ad = InputBox("Aranacak Personelin Adini Girin", "ARAMA")
PBil.git ilk
bulundu = PBil.sontutanak
While Not bulundu
```

```
If InStr(1, PBil.ad_soyad, ad) > 0 Then
bulundu = True
Else
PBil.git birsonraki
If PBil.sontutanak Then
If InStr(1, PBil.ad_soyad, ad) = 0 Then _
MsgBox "Personel BULUNAMADI", vbExclamation
bulundu = True
End If
End If
Wend
VeriAl
Exit Sub
phata:
MsgBox Err.Description, vbExclamation
End Sub
```

Bu kodda görüleceği gibi clsPBil sınıfından yaratılmış olan Pbil nesnesinin **sontutanak** adlı bir özelliği kullanılmıştır. Bu özellik için clsPBil sınıfı içinde yapılan tanım aşağıdaki gibidir.

```
Public Property Get sontutanak() As Boolean
sontutanak = rs.EOF
If Not rs.EOF Then
rs.MoveNext
sontutanak = rs.EOF
rs.MovePrevious
End If
 End Property
```

Bu özellik, sadece okunabilir bir özelliktir. Kullanım amacı, veritabanının son tutanağına gelinmişse bunu çağrıldığı yere döndürmektir.

Program çalıştığında aşağıdaki gibi bir görüntü çıkacaktir.

Bu form görüntüsünden BUL butonuna basıldığında isim girmeyi sağlayan bir form görüntüsü gelir. Bu form InputBox fonksiyonu kullanılarak çıkarılabilen hazır bir veri alma formudur.

Bu programdaki arama işlemi sadece AD_SOYAD alanı üzerinden yapılmıştır. Aynı işlemleri diğer alanlar için de yapmak mümkündür.

Veritabanındaki tutanakların hepsine kod aracılığıyla bakmak yerine arama işlemi için SQL sorgu dilini kullanmak daha basittir. Şimdi veritabanı üzerinde arama yapmak için SQL kullanımını örnekleyelim. Bunun için yukarıdaki örnekte yapılması gereken değişiklikler adımlar halinde aşağıda gösterilmiştir.

1- frmPBil formunun cmdBul_Click yordamını aşağıdaki gibi değiştirin.

```
Private Sub cmdBul_Click()
Dim ad As String
Dim strSQL As String
On Error GoTo phata
```

```
ad = InputBox("Aranacak Personelin Adini Girin", "ARAMA")
If ad <> "" Then _
strSQL = "SELECT * from personel_bil WHERE ad_soyad LIKE '*" _
& ad & "*';"
frmAra.ara "ad_soyad", strSQL, Me
Exit Sub
phata:
MsgBox Err.Description, vbExclamation
End Sub
```

2- Burada kullanılan frmAra formu sadece ListView nesnesi içeren bir formdur. Projeye bir form ekleyip bu formu aşağıda belirtildiği gibi yaratın.

Form	Name	frmAra
	Caption	SORGU SONUÇLARI
Listview	Name	lstSonuc
	View	3- lvwReport
	LabelEdit	1-lvwManual

Bu formun içine aşağıdaki yordamı kodlayın.

```
Public Sub ara(alan As String, _
strSQL As String, frm As Form)
Dim PBil As clsPBil
Dim rs As Recordset
Dim fld As Field
Dim kno As Long
Set PBil = New clsPBil
```

```
Set rs = PBil.ara(strSQL)
If Not rs.EOF Then
lvwSonuc.ColumnHeaders.Add , "kno", _
"TUTANAK", 700
For Each fld In rs.Fields
lvwSonuc.ColumnHeaders.Add , fld.Name, _
fld.Name, 200 * Len(fld.Name)
Next
Do
kno = kno + 1
lvwSonuc.ListItems.Add kno, , CStr(kno)
For Each fld In rs.Fields
lvwSonuc.ListItems(kno). _
SubItems(fld.OrdinalPosition + 1) = _
fld.Value & " "
Next
rs.MoveNext
Loop While Not rs.EOF
rs.Close
Set rs = Nothing
Me.Show vbModal, frm
Else
MsgBox "TUTANAK BULUNAMADI", vbInformation
Me.Hide
End If
End Sub
```

Bu yordamda önce Pbil adlı clsPBil sınıfından bir nesne üretiliyor. Daha sonra bu nesnenin **ara** adlı fonksiyonu çağrılarak sorgu sonuçları bir RecordSet yapısı içine alınıyor.

3- clsPBil sınıfının yeni fonksiyonu **ara** için aşağıdaki kodu bu sınıfın koduna ekleyin.

```
Public Function ara(sSQL As String) As Recordset
Set ara = db.OpenRecordset(sSQL, dbOpenDynaset, _
dbReadOnly)
End Function
```

Bu kodda da görülebileceği gibi Class_Initialize yordamıyla açılan ve **db** veritabanı değişkeni kullanılarak veritabanı üzerinde bir sorgu yapılıyor. Sorgu sonuçları bu fonksiyon yardımıyla fonksiyonun çağrıldığı yerdeki bir RecordSet yapısı içine döndürülüyor. Sorgu sonuçları için sadece okunabilir, Dynaset yapısında bir Recordset kullanılıyor.

Programın çalıştırılıp BUL butonuna basıldığında aşağıdaki ekran görüntüsü belirecektir.

TUTANAK	sicno	ad_soyad	dtar	dyer
1	a5206	Fatih Turkmen	01.07.1968	Manisa
2	b9622	Feride Limon	06.02.1969	Kahraman...
3	a2237	Fadime Çiçek	12.02.1962	Adana
4	c4646	Mehmet Ali Çakir	02.02.1961	Ankara

Bu sonuç isminin içinde "**i**" harfi geçen kişilerin listesidir. Bunun SQL dilindeki ifadesi şu şekilde olur.

SELECT * FROM personel_bil WHERE ad_soyad LIKE '*i*'

Bu sorgu clsPBil sınıfının **ara** fonksiyonunda veritabanına OpenRecordset metodu aracılığıyla verilmekte ve veritabanı da bu sorgu sonucunu bir Recordset yapısı içinde döndürmektedir.

Veritabanı üzerinde arama yapmanın başka bir yolu da indeks kullanmaktır. Indeks kullanmak için Recordset yapısının **Seek** metodu kullanılır.

1- Seek metodu sadece tablo türü Recordset yapılarında kullanıldığı için ilk yapılması gereken clsPBil sınıfının Class_Initialize yordamında OpenRecordset metodunun paramatrelerini değiştirmektir. Bu metodun kullanılışını aşağıdaki gibi değiştirin.

Set rs = db.OpenRecordset("personel_bil", dbOpenTable, _

dbSeeChanges, dbOptimistic)

2- Bu Recordset yapısı için indeks belirlemek **index** özelliğine değer aktarmakla olur.

Bu işlemi gerçekleştirmek için aşağıdaki kodu da Class_Initialize yordamına ekleyin.

rs.Index = "ind_sicno"

3- Aşağıdaki yordamı clsPBil sınıfına ekleyin

```
Public Sub indeksAra(deger As String)
Dim aktiftut As Variant
aktiftut = rs.Bookmark
rs.Seek "=", deger
If Not rs.NoMatch Then
TutanakOku
Else
rs.Bookmark = aktiftut
HataVer tutanakyok
End If
End Sub
```

Burada önce aktif tutanağın konumu saklanmakta, Recordset yapısının **Seek** metodu çağrılıyor. Eğer tutanak yoksa aktif tutanağa tekrar dönülüyor.

4- frmPBil formunun cmdBul_Click yordamını aşağıdaki gibi değiştirin

```
Private Sub cmdBul_Click()
Dim ad As String
Dim strSQL As String
On Error GoTo phata
ad = InputBox("Aranacak Personelin Sicil " & _
"Numarasını Girin", "ARAMA")
If ad <> "" Then PBil.indeksAra ad
VeriAl
Exit Sub
phata:
MsgBox Err.Description, vbExclamation
End Sub
```

Bu program sicil numarası verilen personelin bilgilerini formda görüntüler.