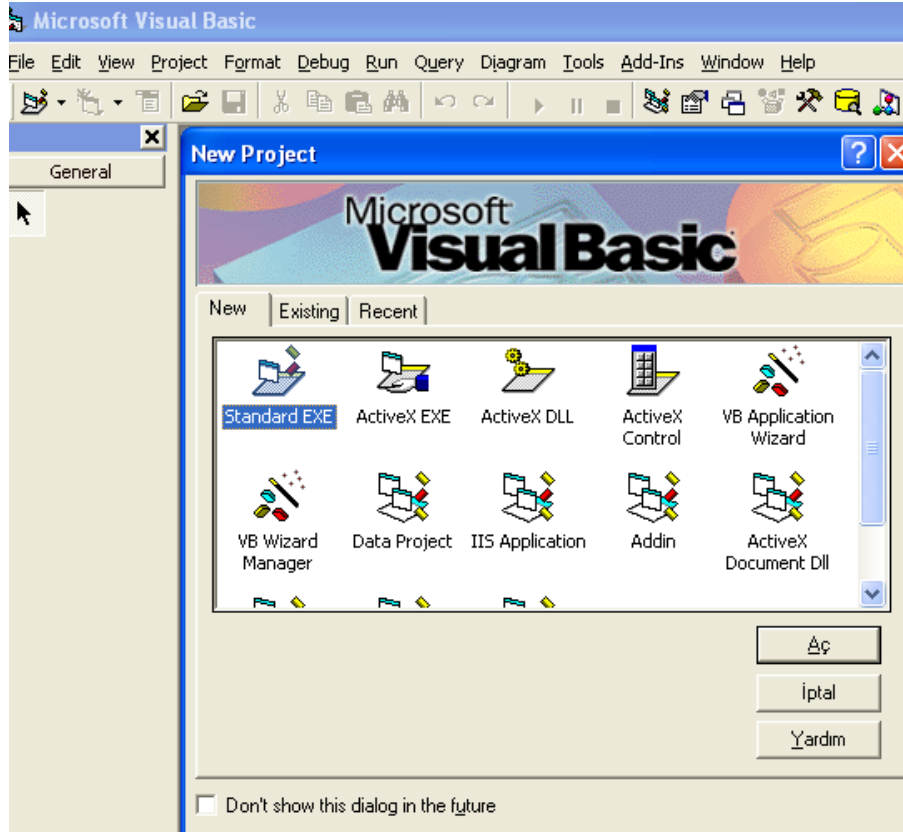


2- VISUALBASIC

Visual Basic nesne tabanlı (object oriented) görsel bir programlama dilidir. Öğrenilmesi, uygulanması oldukça kolaydır. Bu nedenle üniversitemizin Endüstriyel Elektronik programında ilk öğretilecek programlama dili olarak seçilmiştir.

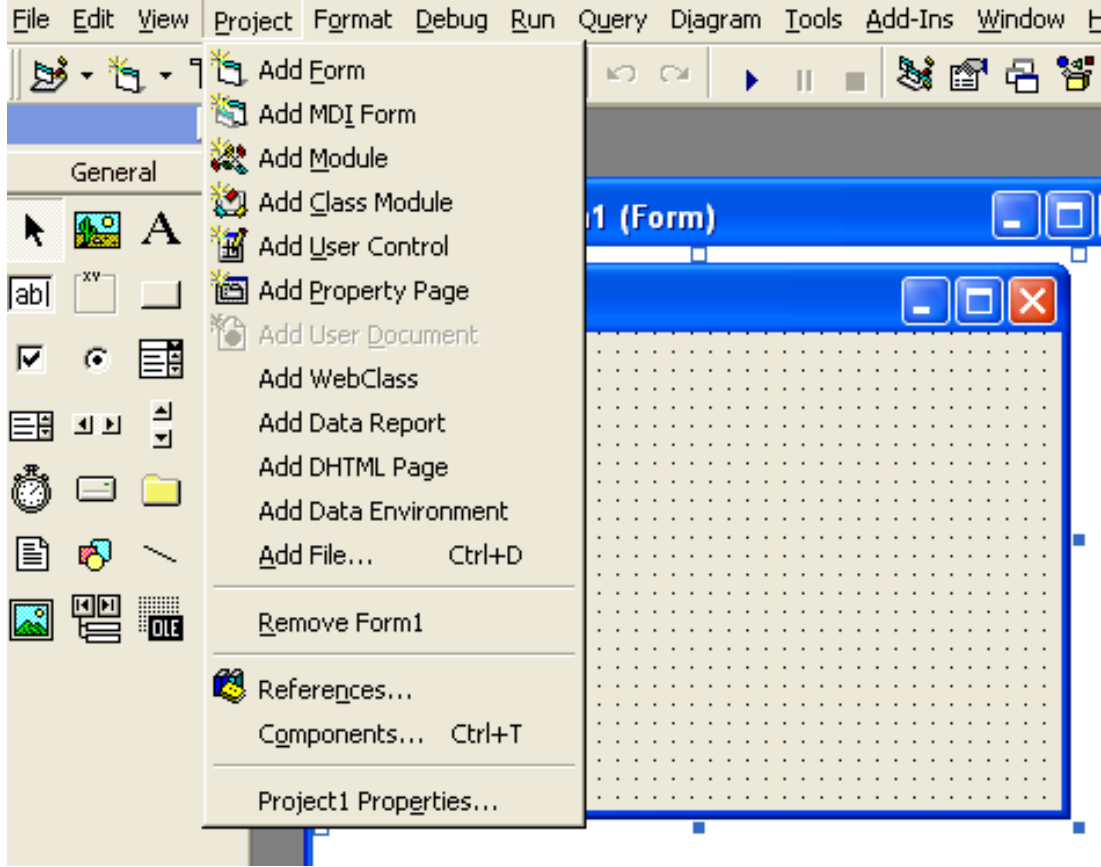
2-1 Visual BASIC 6.0 PROJE TASARIM PENCERESİNE BAKIŞ

Visual BASIC 6.0 açıldığı zaman karşınıza Şekil 2.1'de görülen ekran görüntüsü gelecektir.



Visual Basic açılış ekranı

Örnekler Standard EXE projeleri ile oluşturulacaktır. Visual basic tasarım penceresi açıldığında göze ilk çarpan önemli özellik tasarım ortamının MDI modunda açılmasıdır.(Şekil 2.2)



Visual BASIC 6.0 Proje Tasarım Penceresi

MENÜ SATIRI:

11 adet ana menü başlığı ve herbirinin alt menülerini içerir. Visual BASIC ortamında proje(yazılım) geliştirirken yararlanılabilecek çeşitli fonksiyonları vardır.

KONTROL NESNELERİ:

Visual BASIC ortamına önceden yerleştirilmiş nesnelerdir. Bu kontrollerden gerekli olanları form üzerine taşınır ve yazılım icrası esnasında istenilen fonksiyonları yerine getirebilmeleri için bu kontrollere bağlı bilgisayar programları(kodlar/codes) da oluşturulur.

FORM:

Visual basic pojesinin temel nesnesidir. Proje ile ilişkili tüm nesneler form üzerine yerleştirilir. Bir projede, gerekiyorsa birden fazla form da bulunabilir.

PROJECT:

O anda aktif formla ilişkili olarak bazı işlevleri yerine getirir. Pencere başlığının hemen altında iki guruba ayrılmış üç buton bulunur. Bunlardan View code butonu seçilirse o esnada aktif olan nesneye bağlı olarak oluşturulmuş bilgisayar programı(kodu) görüntülenir. View Object butonu seçilirse de, bu durumda tasarım aşamasında o an için seçilmiş olan nesne x görüntülenecektir.

Tolgggle Folders butonu ile ise proje içinde kullanılan genel maksatlı modüller, ormlar, inıflar ve kaynak dosyaları gibi proje bileşenleri bir dizin mantığı içinde ayrıntılı veya toplu halde görüntülenir.

ÖZELLİKLER :

o anda aktif durumdaki (seçilmiş) Visual BASIC kontrolünün tüm özellikleri görüntülenir.

IMMEDIATE:

breake modda (tasarım ortamında program akışının kesilmesi) otomatik olarak boş olarak açılır.Debug,reset ve komut satırındaki işlemler uygulanabilir veya program akışını kalındığı yerden devam edilebilir.

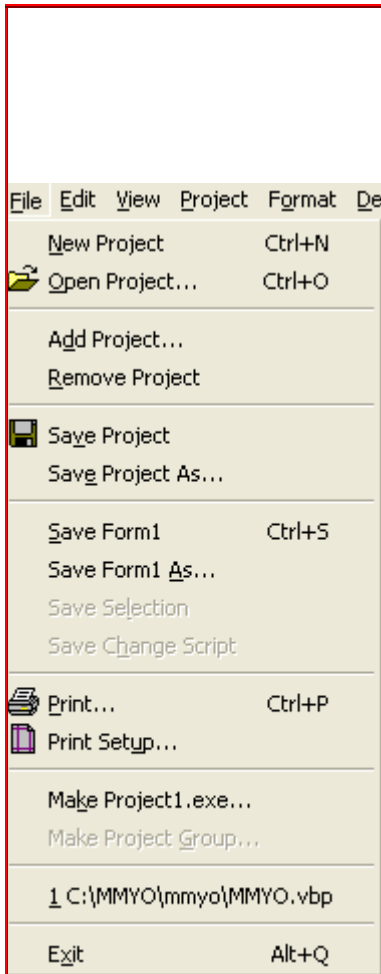
LAYOUT:

Formların ekrandaki pozisyonlarını mouse yardımı ile ayarlanabilir.

2-1-1 MENÜ SATIRI

FILE (DOSYA) MENÜSÜ:

Şekil deki alt menülerden oluşur.



New project:Yeni bir Visual BASIC projesi başlatmak amacı ile kullanılır.

Open Project:Daha önceden diskte saklamış olduğunuz bir VB uygulamasını tekrar aktif hale getirir.

Add Project:Proje penceresi birden fazla proje içerdiğinde ikinci bir projeyi ekler.

Remove Project:Aktif uygulamamız içindeki bir projeyi uygulama içinden çıkarmak için kullanılır.

Save Project Group:O esnada uygulamada kullanılan proje ve projelerle ilişkili tüm dosyaları ve ilişkili Visual BASIC bilgilerini tek bir isim altında saklamak amacı ile kullanılır. Bir kez kayıt edilince group yazısı bir daha gelmez.

Save Project Group As:Aktif çalışılan bir projeyi son hali ile diskten farklı tüm uygulama bileşenlerini yeni bir isimle saklamayı sağlar.

Save (Form1)Dosya İsmi :Sadece o esnada aktif proje penceresinde seçilmiş olan bir nesneyi saklar.

Save (Form1)Dosya İsmi As: Daha önce saklanmış olan bir dosya yeni bir isimle yeniden saklanabilir.

Print:Project ile ilgili istenilen (kod sayfasını,formu) bölümleri yazıcıya gönderir.

Print Setup:Uygulamanızı yazıcıya aktarma durumunda yazıcı tipini, yazıcıdaki kâğıt baskısını,yatay,dikey durumunun ayarı.

Make Project..exe:Çalıştırdığımız Visual Basic uygulamasını EXE haline çevirerek Visual BASIC ortamından bağımsız çalışılmasını sağlar.

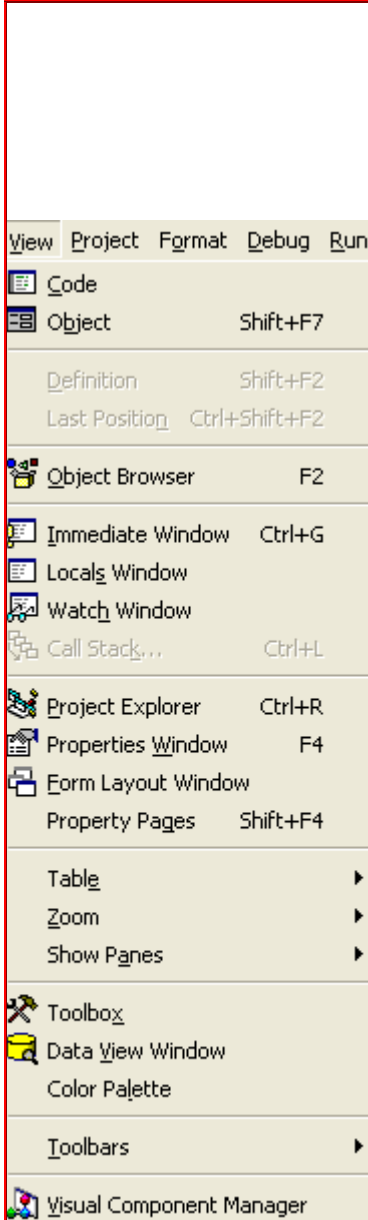
Make Project Group:Uygulamamız birden fazla project içeriyorsa görüntülenir. Uygulamanızda oluşturmak istediğiniz projeleri seçmenizi sağlar.

Exit : Çıkış

EDIT (DÜZEN) MENÜSÜ

	Undo Delete(Gerial): Yapılan son işlemi geri alır. Can't Redo(Yinele): Geri alınan işlemi tersine çevirir. Cut(Kes): Seçilen alanı silip panoya alır. Copy(Kopyala): Seçilen alanı panoya kopyalar. Paste(Yapıştır): Panodaki bilgileri yazdırır. Paste Link(Özel Yapıştır): Panodaki bilgileri istenilen biçimde yapıştırır. Delete: Seçilmiş metin yada nesneyi siler. Select All: Aktif kod penceresindeki bütün kodu veya form üzerindeki bütün kontrolleri seçmek için kullanılır. Find(Ara): Visual BASIC kodu içerisinde bir metni aramak için kullanılır. Find Next(Aramaya Devam): Find ile yapılan arama işlemine kaldığı yerden devam etmek için kullanılır. Replace(Değiştir): Bir metin içinde bir sözcüğü aratıp, bulunduğu takdirde başka sözcükle değiştirmek. Indent(Tab): Tab tuşunun görevini yapar. Outdent(Shift Tab): Tab tuşunun ilerlettiği sütun miktarı kadar sola kaydırır. Insert File(Dosya Birleştir): Çalışılan kod sayfasının içerisine başka bir dosyada bulunan kodu eklemek için kullanılır. List properties/Methods: Kod penceresine her uygun ifadenin ardından bu ifadeden sonra yazılabilecek mümkün diğer ifadeleri içeren bir yardım kutucuğu görüntüler.(data1. sonra gelecek menü) List Constants: Kod penceresinde,yazılan ifadenin alabileceği sabit değerleri içeren yardımcı bir pencere görüntüler. Quick Info: Kod penceresinde seçilen fonksiyonların,metodların,prosedürlerin ve değişkenlerin yazılımını gösteren yardımcı bir pencere görüntüler.. Parameter Info: Kod penceresinde yazılan veya ifadelerin ,çerdikleri parametreleri gösteren yardımcı bir pencere görüntüler. Complate Word: VB, yazmakta olduğunuz ifadeden kelimeyi bulmaya çalışır. Ve kullanmanızı sağlar. Go to Row: Satıra git (Birinci,Önceki,Sonraki,Yeni) Bookmark: Program kodunun çok sık kullanıldığı satırlarına kolay bir şekilde ulaşmak için kullanılır.
 Edit	
 Undo Typing	Ctrl+Z
 Can't Redo	
 Cut	Ctrl+X
 Copy	Ctrl+C
 Paste	Ctrl+V
 Paste Link	
 Remove	
 Delete	Del
 Delete Table from Database	
 Select All	Ctrl+A
 Select All Columns	
 Table	
 Find...	Ctrl+F
 Find Next	F3
 Replace...	Ctrl+H
 Indent	Tab
 Outdent	Shift+Tab
 Insert File...	
 List Properties/Methods	Ctrl+J
 List Constants	Ctrl+Shift+J
 Quick Info	Ctrl+I
 Parameter Info	Ctrl+Shift+I

VIEW (GÖRÜNÜM) MENÜSÜ



Code: O esnada seçilmiş olan form ya da modülün Code Penceresini(prg. Yazıldığı pencere)görmek amacı ile kullanılır.

Object: Aktif kod ile ilişkili olan nesneyi(form gibi)görmek için kullanılır.

Definition: Form yada modülün Code Penceresinde kursorün bulunduğu aktif prosedüre ait kod penceresini görüntüler.(Recordset,Data1)

Last Position: Kod penceresinde en son noktaya yeniden sıçrama yapmak istediğinizde kullanılır.

Object Browser: Bu seçenek ile Object Browser(Nesne Gözetici)penceresi ekrana gelir.

Immediate Window: Bu menü alternatifi ile Immediate (hata ayıklama penceresi) ekrana gelecektir.

Locals Window: O esnada aktif prosedürde ki bütün değişkenleri ve değerlerini gösterir.

Watch Window: B-Break Modda, daha önceden gözlem amacı ile tanımlanmış değişken ya da ifadenin o andaki değerini görmek için kullanılır.

Calc Stack: Projeniz içinde, o esnada aktif olan prosedür çağrılarının listesini gönderir.

Project Explorer: Project (proje) başlıklı pencereyi ekrana getirir.

Properties Window: Properties(Özellikler) başlıklı pencereyi ekrana getirir. Bu pencerede o esnada aktif olan nesne için geçerli olan özelliklerin bir listesini ve bu özelliklere o esnada atanmış olan değerleri incelemek mümkündür.

Form Layout Window: Daha öncede söylendiği gibi Layout penceresinde Formların ekrandaki pozisyonları mouse yardımı ile ayarlanabilir.

Property Pages: Tasarım ortamında kullanıcı tarafından oluşturulmuş bir kontrolün özelliklerinde değişiklik yapmak amacı ile özellik sayfalarını listeler.

Table

Zoom

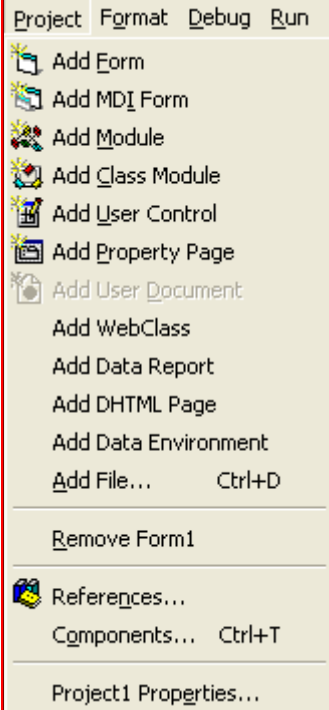
Show Panes

Toolbox: Araç kutusunu görünür hale getirir.

Color Palette: Seçilmiş Visual BASIC nesnelerinin renklerini değiştirmek üzere, renk paletini ekrana getirir.

Toolbar: Visual BASIC kullandığı araç çubuğu gruplarından biri ekranda görüntülenebilir veya silinebilir.

PROJECT MENÜSÜ



Add Form: Aktif projenin içine yeni veya mevcut bir form katmak için kullanılır. (Form1 varken Form 2 ye geçilebilir.)

Add MDI Form: Bu menü alternatifi ile bir Multiple Document Interface (çok sayıda Doküman İçeren Arayüz) oluşturulabilir veya mevcut bir MDI formu uygulamaya ekleyebilirsiniz.

Add Module: Visual Basic'in bir forma ya da nesneye bağlı olmayan ve uygulamanın tüm bileşenleri tarafından kullanılabilen kodların modul adı verilmektedir.

Add Class Module: Bu menü alternatifi ile Sınıf Modülü (Class Module) denilen özel tipte bir modül oluşturmak mümkün olmaktadır. Sınıf modülünde, projenin tüm bileşenleri tarafından kullanılacak bir sınıf ile ilişkili tanımlar bulunabilir.

Add User Control: Sadece kendi projenizde kullanabileceğiniz ActiveX kontrolleri oluşturmak için kullanılır. Active X kontrollerini başka tasarım ortamlarında da kullanmak için projenizi ActiveXControl olarak başlatmanız gerekir.

Add Property Page: Oluşturduğunuz ActiveX kontrolleri için Özellik sayfası isimlendireceğimiz Property Page/ler oluşturmak için kullanılır.

Add User Document: Bu menu komutu Standart EXE modülü dışındaki proje modülleri ile beraber kullanılır. Projeye yeni yeni bir ActiveX dökümanı eklemek için kullanılır.

Add File: Daha önceden hazırlanmış genel maksatlı modüller, formlar, sınıf modülleri ve kaynak dosyalar gibi bileşenleri, o esnada aktif olan projenizin içine katma imkanı sağlar.


Remove (Dosya Adı)Form..(1,2,3) : Aktif projenizin içindeki bir dosyayı bu proje içinden çıkarma imkanı sağlar. Çıkarılan dosya disketten silinmez.

References: Projenizin içerisine başka uygulama programları ya da nesne kütüphanelerinden nesneler katma imkanı sağlar.

Components: Visual BASIC'te kullanabileceğimiz kontrol nesnelerinin sayı ve çeşidini arttırmak ve DLL uzantılı tasarımları projeye eklemek amacı ile kullanılır.

Project1 Properties: Visual BASIC projesi ile ilgili düzenlemeler yapmak için kullanılır. Diyalog penceresinin BAŞLIK ÇUBUĞU, Project Explorer penceresinde seçili aktif projenin ismini gösterir.

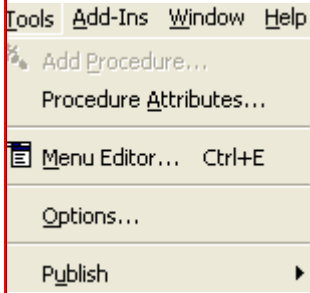
FORMAT (BİÇİM) MENÜSÜ

	<p>Align: Form üzerinde seçilmiş nesnelerin pozisyonları, enson seçilen nesneyi referans alacak şekilde aşağıdaki alternatifler baz alınarak hizalanır.</p> <p>Left sol, Centers ortala, Tops Üst köşe, Middles üst alt orta noktaları eşit, Bottoms Alt köşe, To Grid en yakın Grid üst köşe noktası</p> <p>Make Same Size: Bu menü alternatifi ile seçilen nesnelerin boyutlarında ayarlama yapılır. Width Genişlik, Height Yükseklik, Both seçilen nesnelerin boyutları referans alınan nesne ile aynı yapılır</p> <p>Size To Grid: Seçilen nesneleri otomatik olarak formun en yakın grid noktalarına yerleştirir.</p> <p>Horizontal Spacing: Seçilen nesnelerin kendi aralarındaki yatay boşluklar istenilen alternatiflere göre ayarlanır.</p> <p>Make equal: Seçilen nesnelerin kendi aralarındaki yatay boşlukları eşit hale getirir. Increase: Seçilen nesnelerin kendi aralarındaki yatay boşluğu, kabul edilen bir grid hücresinin eni kadar artırır.</p> <p>Decrease: Seçilen nesnelerin kendi aralarındaki yatay boşluğu kabul edilen bir grid hücresinin eni kadar azaltır.</p> <p>Remove: Seçilen hücrelerin kendi aralarındaki yatay boşlukları kaldırır.</p> <p>Vertical Spacing: Yukarıdaki menü seçeneğine eşdeğer bir fonksiyonu vardır. Sadece boşluk ayarlamaları düşey noktada yapılır.</p> <p>Center in Form: Seçilmiş nesne grubu, bu grubun içindeki en dış nesnelere göre form üzerinde ortalanır.</p> <p>Horizontaly: Seçilmiş nesne grubu içinde en sol dışdaki nesne ile en sağ dışdaki nesnenin, sırası ile üst ve alt köşe noktalarına göre tüm seçilmiş nesneleri yatay olarak ortalar.</p> <p>Vertically: Seçilmiş nesne grubu içinde en üst dışdaki nesne ile en alt dıştaki nesnenin, sırası ile üst ve alt köşe noktalarına göre bir ayarlama yapılır.</p> <p>Order: Order menu seçeneği ikiye ayrılır.</p> <p>Bring to Front: Seçilmiş olan nesneyi, formda mevcut diğer nesnelerin önüne getirir.</p> <p>Send to Back: Seçilmiş olan nesneyi, diğer nesnelerin arkasına yerleştirir.</p> <p>Lock Controls: Form üzerindeki kontrolleri, bulundukları pozisyonda kilitler. Öyle ki, kontroller bu komuttan sonra form üzerinde hareket ettirilemez.</p>
---	---

DEBUG MENÜSÜ

	<p>Step Into: Tasarım esnasında Visual BASIC kodundaki o esnada kursorün İşaret etmiş olduğu komutu çalıştırır.</p> <p>Step Over: Step info komutuna benzer. Farkı, icra edilen aktif komutu, bir prosedür çağırma komutu ise, o prosedür tek bir komut gibi bütünü ile icra edildikten sonra, esas koddaki bir sonraki deyme geçilir.</p> <p>Step Out: O esnada icra edilen fonksiyonun içinde kursorün işaret etmiş olduğu komuttan itibaren bütün komutlar çalıştırılır ve fonksiyonun çağırıldığı esas kodda bir sonraki deyme geçilir.</p> <p>Run To Cursor: Gene sadece tasarım anında kullanılabilecek bir komuttur. Programın icrası durmuş iken(Break Mode) icrayı yeniden başlatarak bir sonraki durma noktasının hangi komut olacağını belirler.</p> <p>Add Watch: Bir değişken yada ifadenin programın icrası esnasında hangi değerleri aldığını izlemek üzere bir gözlemci mekanizma kurar.</p> <p>Edit Watch: Gözlemci tanımı ile ilişkili değişiklik ya da düzeltme yapmak gerektiğinde kullanılır.</p> <p>Quick Watch: Brake modda üzerinde durduğu değişkenin, özelliğın veya başka bir ifadenin değerini gösteren bir diyalog penceresi görüntüler.</p> <p>Toggle Breakpoint: Tasarım esnasında kullanılabilen bir komuttur. Kod üzerinde ki durma noktalarını belirlemek için kullanılır.</p> <p>Clear All Breakpoints: Toggle breakpoint komutu ile yerleştirilmiş olan bütün durma noktalarını iptal eder.</p> <p>Set Next Statement: Sadece Break modunda kullanılabilir. Break moduna şu durumlarda erişilir.</p> <p>Show Next Statement: Gene sadece break modda kullanılabilecek bir komuttur. İcra edilecek bir sonraki deymiy gösterilir.</p>
	<p>Start: Geliştirdiğimiz projeyi icra etmek amacı ile kullanılır.F5 kullanılır.</p> <p>Start With Full Compile: Start komutu ile Visual Basic sadece aktif kodu ve ilişkili kısımları derler. Belli bir anda projenizin tümünün derlenerek çalıştırılması istiyorsanız bu alternatifi seçmelisiniz.</p> <p>Break: Normal akışını sürdüren programı durdurur ve programın çalışmasını breake modda sürdürölür.</p> <p>End: İcra edilen programı durdurarak Visual BASIC'in kullandığı tüm sistem kaynaklarını iade eder.</p> <p>Restart: İcrası durdurulmuş olan programın yeniden icra edilmesini sağlar.</p>

TOOLS MENÜSÜ



Add Procedure: Kod penceresi içine bir prosedür yerleştirmek amacı ile kullanılır.

Procedure Attributes: Belirtilen herbir özellik veya metod için ayarlamalar yapabileceğiniz veya mevcut attribute'leri(nitelikleri) gözlemleyebileceği bir diyalog penceresi görüntüler.

Menu Editor: Geliştirdiğiniz uygulama içinde Visual BASIC ve Windowsun menülerine benzer şekilde menüler oluşturmak istediğinizde Menü Editörü size büyük kolaylık sağlayacaktır.

Options: Bu menü seçeneği kullanılırsa Visual BASIC ortamında çalışırken mevcut olan çeşitli sistem parametrelerini değiştirmek imkanı sağlayan diyalog penceresi karşınıza gelecektir.

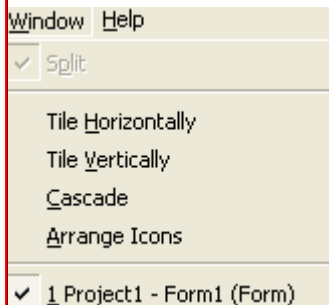
ADD-INS MENÜSÜ



Visual Data Manager: Bu menü alternatif, Visual BASIC içine, veri tabanlarının yönetimi konusunda çok yetenekli bir yazılım modülü olan Visual Data Manager'ın eklenmesini sağlar.

Add-In Manager: Bağımsız yazılım modüllerinin Visual Basic ortamına katılmasını ya da bu ortamdan çıkarılmasını sağlar.

WINDOW MENÜSÜ



Split: Sadece kod penceresi(programın yazıldığı pencere)açık iken aktif olur. Kod penceresini yatay olarak ikiye ayırır.Eski hali için split seçilir.

Tile Horizontally: MDI modda,açık pencereleri yatay olacak şekilde eşit pencere boyutlarında ekrana yerleştirir.

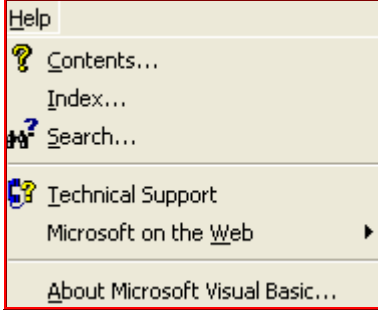
Tile Vertically: Yukarıdaki menü seçeneği ile eşdeğerdir. Farklı olarak pencere ayarlamaları düşey olacak şekilde gerçekleştirilir.

Cascade: MDI modda, açık pencerelerin ekrandaki yerlerini üst üste cascade bir tarzda yeniden ayarlar.

Arrange Icons: MDI modda minimize edilmiş pencere ikonlarını anapencerenin alt sol köşesine göre yeniden düzenler.

WindowList: Bütün açık pencereleri listeler. Aktif yapılmak istenen pencere seçilirse ilgili pencere isminin sol yanında o esnada aktif pencereyi gösteren işareti gözükecektir.

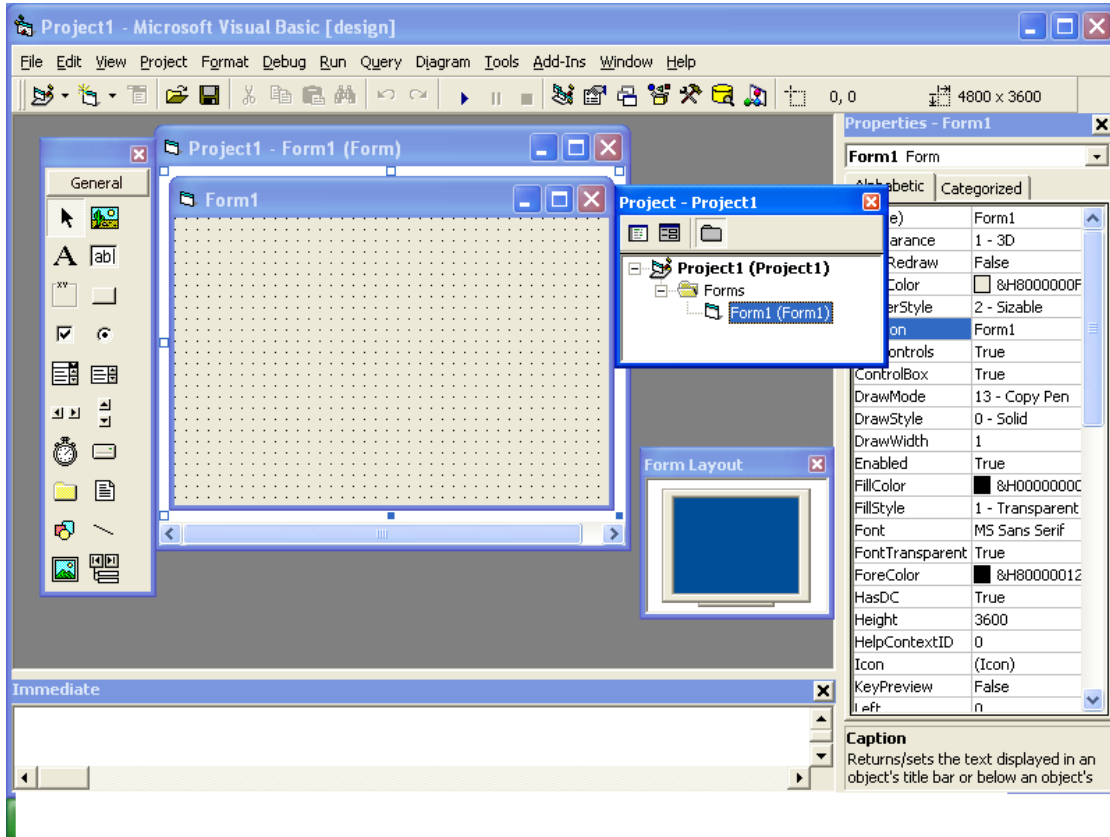
HELP MENÜSÜ



Visual BASIC konuları hakkında yardımcı bilgiler sunar.

2-1-2 PROJELER VE BİLEŞENLERİ

Visual Basic'te bir uygulama geliştirmek için proje denilen yapı içerisinde gerçekleştirilir. O halde yeni uygulama yeni bir proje oluşturmayı gerekli kılacaktır. Bir Visual BASIC Projesi, formlar, formlar üzerine yerleştirilen kontrol nesneleri, nesnelere bağlı kodlar (event procedures-olay prosedürleri) ile bağımsız modüllerden oluşur. Visual BASIC 'te proje dosyalarının uzantıları .max ya da .vbp şeklindedir.



Başlangıç Penceresi

Şekilde görülen başlık satırı o anda aktif olan projenin adını göstermektedir.

Menü Satırı:

11 adet ana menü başlığı ve herbirinin alt menülerini içerir. Visual BASIC ortamında proje(yazılım) geliştirirken yararlanılabilecek çeşitli fonksiyonları vardır.

Araçlar Satırı:

Menü içerisinde yer alan bazı işlemleri kullanıcının daha hızlı gerçekleştirebilmesi için kullanılabileceği ikonları içerir. Bu ikonlardan herhangi biri üzerine götütürseniz mouse'yi bu ikonun gerçekleştireceği menü seçeneğinin ne olduğunu görebilirsiniz.

Özellikler Penceresi:

O esnada ilgi odağı olan (seçilmiş, aktif) nesneye ait özellikler ve o andaki değerlerini listeler. Ekranda bu pencere yoksa **View/Properties Window** menü adımları ile çağırılabilir.

Proje Penceresi:

O esnada aktif durumda olan proje ile ilişkili dosyaların listesini görüntüler. Bu dosyalar, projeye ait nesneler (form, kontroller vb) ya da bağımsız modüllere ait dosyalardır. Proje penceresi o esnada yoksa View/Properties Explorer menü adımları ile çağırılabilir.

Araçlar Kutusu:

Visual BASİC uygulamasını geliştirirken oluşturduğunuz projede kullanılabileceğiniz Visual BASİC kontrol nesnelerini içermektedir. Eğer yoksa **view /Toolbox** menü adımları ile onu ekrana getirebilirsiniz.

ÖZELLİKLER PENCERESİ

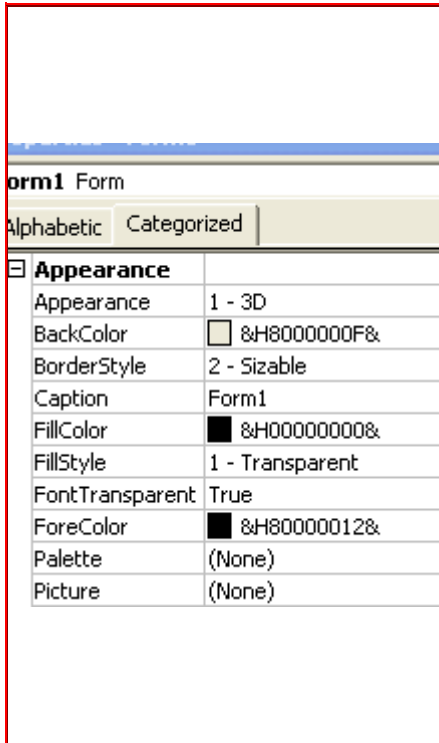
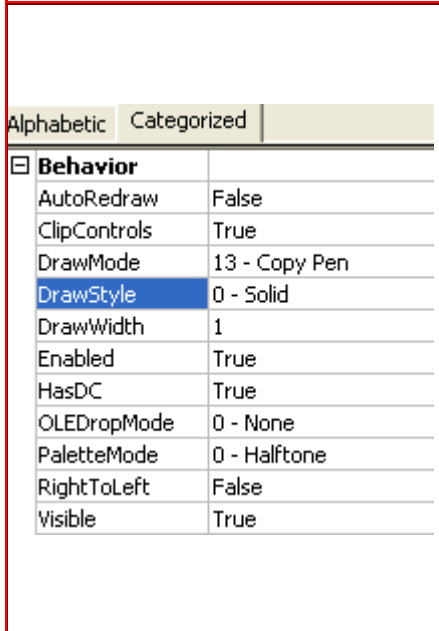
Özellikler:

Penceresi, o esnada ilgi odağı olan (seçilmiş,aktif) nesnenin sahip olduğu özellikleri ve bu özelliklerin o andaki değerlerini görüntüler. Özellikler penceresini **View /Properties Window** menü yolu ile ya da **F4** tuşuna basarak ekrana getirmek mümkündür.

Aşağıdaki özellikler penceresinin başlığı Form1'dir. Form1 başlık satırının bir alt satırında ise Form1 Form ibaresi görülmektedir.Bunun anlamı başlıklı nesnenin Form türünde bir nesne olduğudur. Özellikler penceresi bir kontrolün özelliklerini iki farklı diziliş sırasında listeleyebilir. Bunlardan ilki Alfabetic seçeneğidir. Adındanda

anlaşılacağı gibi penceredeki özellikleri alfabetik olarak takip etme imkanı sağlar Categorized seçeneği ise özellikleri, Appearance (görünüş) behavior (davranış), DDE, FONT vb. bazı ortak gruplar altında listelemek için kullanılır. Özellikler penceresine dikkat edilirse, iki sütuna sahip olduğunu göreceksiniz. Bu sütunlardan soldaki özelliklerin adlarını (Caption, name, height vb), sağdaki ise bu özelliklerin o esnada sahip olduğu değerleri göstermektedir.

2-1-3 PROPERTIES (ÖZELLİKLER PENCERESİ)

 <p>Form1 Form</p> <p>Alphabetic Categorized</p> <p>Appearance</p> <table><tr><td>Appearance</td><td>1 - 3D</td></tr><tr><td>BackColor</td><td>&H80000000F&</td></tr><tr><td>BorderStyle</td><td>2 - Sizable</td></tr><tr><td>Caption</td><td>Form1</td></tr><tr><td>FillColor</td><td>&H00000000&</td></tr><tr><td>FillStyle</td><td>1 - Transparent</td></tr><tr><td>FontTransparent</td><td>True</td></tr><tr><td>ForeColor</td><td>&H800000012&</td></tr><tr><td>Palette</td><td>(None)</td></tr><tr><td>Picture</td><td>(None)</td></tr></table>	Appearance	1 - 3D	BackColor	&H80000000F&	BorderStyle	2 - Sizable	Caption	Form1	FillColor	&H00000000&	FillStyle	1 - Transparent	FontTransparent	True	ForeColor	&H800000012&	Palette	(None)	Picture	(None)	<p>Name: Yazılacak program(kod)içinde, forma referans vermek için kullanılan formun adı (Name özelliği) özelliğini verir.(örn bordroform adı verilebilir)</p> <p>Appearance: Form üzerine yerleştirilmiş olan kontrol nesnelerinin 3 boyutlu görünüme sahip olup olmayacaklarını belirler. Önceden atanmış değer 1 'dir. Değer 1 ise 3-Boyutlu görünüme sahip, 0 ise 2-Boyutlu görünüme sahip olacaktır.</p> <p>AutoRedraw:</p> <p>BackColor: Form penceresinin renkli veya standart renkte görünmesini sağlar.</p> <p>BorderStyle: Formun icrası esnasında görünümünü ve bazı fonksiyonlarını belirleyen bir özelliktir.</p> <p>BorderStyle 0 : İcra esnasında formun başlığı mevcut olmayacak formun boyutu mouse ile değiştirilemeyecektir. 1:Kontrol menüsü mevcut olacak, boyut değiştirilemeyecektir. 2: Başlığı görülen kontrol menüsü mevcut ,form boyutu değiştirilebilir olacak. 3: Formun başlığı olacak , kontrol menüsü olmayacak ,boyut değiştirilemeyecektir. 4:Formun başlığı vardır. Kontrol menüsü yoktur ve boyutu sabittir. 5: Windows 95 altında ise Close butonu olan formlar oluşturmak için kullanılır.</p> <p>Caption: Caption özelliği ile formun başlığının değiştirilmesi sağlanır</p>		
Appearance	1 - 3D																						
BackColor	&H80000000F&																						
BorderStyle	2 - Sizable																						
Caption	Form1																						
FillColor	&H00000000&																						
FillStyle	1 - Transparent																						
FontTransparent	True																						
ForeColor	&H800000012&																						
Palette	(None)																						
Picture	(None)																						
 <p>Alphabetic Categorized</p> <p>Behavior</p> <table><tr><td>AutoRedraw</td><td>False</td></tr><tr><td>ClipControls</td><td>True</td></tr><tr><td>DrawMode</td><td>13 - Copy Pen</td></tr><tr><td>DrawStyle</td><td>0 - Solid</td></tr><tr><td>DrawWidth</td><td>1</td></tr><tr><td>Enabled</td><td>True</td></tr><tr><td>HasDC</td><td>True</td></tr><tr><td>OLEDropMode</td><td>0 - None</td></tr><tr><td>PaletteMode</td><td>0 - Halftone</td></tr><tr><td>RightToLeft</td><td>False</td></tr><tr><td>Visible</td><td>True</td></tr></table>	AutoRedraw	False	ClipControls	True	DrawMode	13 - Copy Pen	DrawStyle	0 - Solid	DrawWidth	1	Enabled	True	HasDC	True	OLEDropMode	0 - None	PaletteMode	0 - Halftone	RightToLeft	False	Visible	True	<p>MaxButton - MinButton : Bu değerler mantıksal değerler olabilir. Mantıksal değerler(True-False). Her iki özellikte False durumunda iken boş oluşturulmuş bir formu çalıştırarak Sol üst köşesinden Ekranı kapla ve Simge durumunda küçült seçenekleri olmayacaktır. True yapılırsa tekrar gelecektir.</p> <p>Height: Formun yüksekliğini artırmak ve azaltmak için kullanılır.</p> <p>Width: Bir formun genişliğini saklar. Bir önceki özelliğe tanımlar.</p> <p>Left ve Top: Bu özellikler, formun üst ve sol kenarları ile ekran arasındaki mesafeyi belirler. Top parametresi 0 ise, formun üst kenarı, ekranın üst kenarı ile çakışır Left parametresi 0 is, formun sol kenarı, ekranın sol kenarı ile çakışır.Yükseklik ve genişlik özelliği gibidir.</p>
AutoRedraw	False																						
ClipControls	True																						
DrawMode	13 - Copy Pen																						
DrawStyle	0 - Solid																						
DrawWidth	1																						
Enabled	True																						
HasDC	True																						
OLEDropMode	0 - None																						
PaletteMode	0 - Halftone																						
RightToLeft	False																						
Visible	True																						

Projenin icrası esnasında, formun nasıl görüneceğini belirleyen bir özelliktir. 3 farklı değere sahip olabilir. Değeri 0 ise, bilinen normal form görüntüsü, Değeri 1 ise, formun bir ikona indirgenecektir, Değeri 2 ise form en büyük şeklini alır yani maksimize edilir-WindowState özelliği de, genellikle icra esnasında değiştirilir. Çalışmamız RUN edildiğinde büyüklük ve küçüklük oranı görülecektir.

BackColor: Formun zemin rengini belirleyen özelliktir.

Control Box: Değeri True ise , icra esnasında formun başlık satırının sol kenarında, işletim sistemine özgü bir kontrol menüsü çıkar. False yapılmışsa, icra esnasında bu kontrol menüsü ortaya çıkmaktadır.

Enabled: Özelliği True ise form meydana gelen olaylara duyarlıdır; False ise formun meydana gelen olaylara karşı duyarsız olması sonucu verecektir. Program run edildiğinde mouse ile yapılan hiçbir işlemi kabul etmeyecektir.

Font : Bu özellik ile ilişkili parametreleri ayarlayarak, form üzerinde görüntülenecek yazılara ait fontlar (MS Sans Serif, Courier vb), yazı stilleri (İtalic, bold vb) ve yazı boyutları (size) belirlenebilir.

ForeColor: Bu özellik formun ön-plan rengini belirler. Kullanımı ve renk seçimi Backcolor özelliğindeki gibidir.

Mouse Pointer: İcra esnasında, form üzerinde görüntülenecek mouse göstergesinin (ekran göstergesi) biçimi belirler. Burada 0-16 değerleri alabilir. Burada programın çalışması sırasında mouse görüntüsünü görebilirsiniz.

Mouselcon: Mousepointer de ayarlanan 16 mouse göstergesini artırabilmek için kullanılır. Burada 99'a kadar çıkartılabilir.

Visible: Formun, projenin icrası esnasında, ekranda görünür olup olmayacağını belirler. Değer True ise form icra esnasında görünür, False ise icra esnasında görüntülenmeyecektir.

ScaleMode: Scalemode özelliği, form üzerinde kullanılacak koordinat sistemi için bir birim belirlemek imkanı sağlar. Bu özellik 8 farklı değer alır.

ScaleTop, ScaleLeft: Form üzerine nesneler yerleştirilirken, gözönüne alınacak olan koordinat eksininin başlangıcını belirleyen parametrelerdir.

Scaleheight, Scalewidth: Kullanıcı formu ölçeklemek için kendi birimlerini kullanmak istemesi durumunda kullanılabilecek olan özelliklerdir. Forma eklenecek bir nesne için Form üzerinde bir yatay çizgi belirleyip Properties özelliğinde yükseklik ve genişliğini ayarlarsak nesne o aranda büyük veya küçük olarak forma eklenir.

Alphabetic

Categorized

Misc

(Name)	Form1
ControlBox	True
HelpContextID	0
Icon	(Icon)
KeyPreview	False
MaxButton	True
MDIChild	False
MinButton	True
MouseIcon	(None)
MousePointer	0 - Default
NegotiateMenus	True
ShowInTaskbar	True
Tag	
WhatsThisButton	False

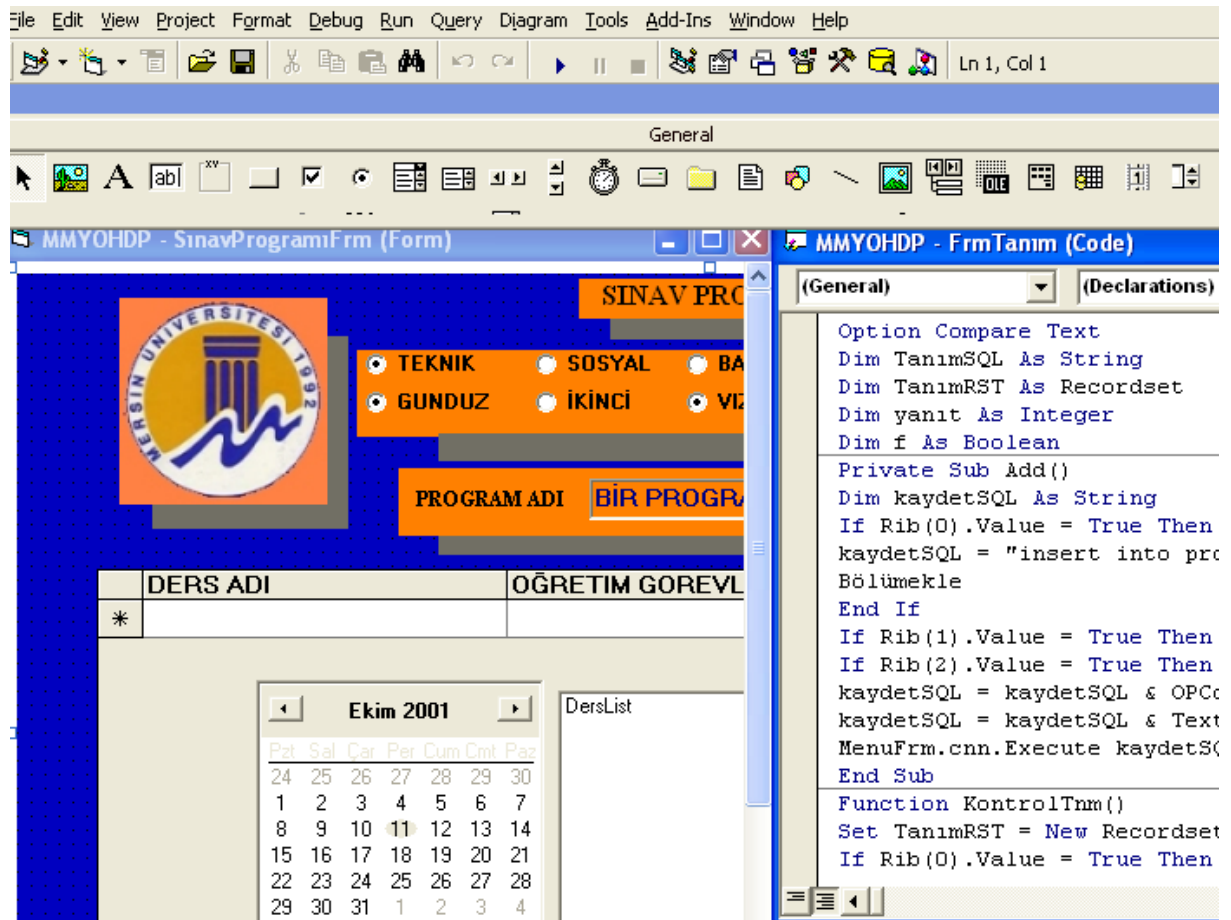
BİR ÖRNEK PROGRAM

Visual Basic'in form penceresini Run ettiğimizde form penceresini mouse ile her tıkladığımızda "Halk Eğitimi Merkezi" yazıp bir alt satırına 'Mouse'u iki kez tıkladın" yazmasını istiyorsak yapılacak işlem şunlardır.

1- Basit bir kod, bir procedure örneği oluşturalım .Form üzerindeyken Mouse'yi iki kez tıkla-tınız.Karşınıza Project isimli bir pencere gelecektir.Load yazan yerin (Proc kutusu)ok İşa-retini tıkladın ve oradakiDbClickseçeneğini seçip işlem satırına yazarız. Aşağıda olduğu gibi

2- Run menüsünden Start'ı seçerek kodu(projeyi)çalıştırınız.

3- Şimdi, ekran göstergesi form üzerinde iken, mouse sol tuşunu iki kez tıkladınız.



2-1-4 KONTROL NESNELERİ

Geliştirilen uygulamanın kullanıcı açısından kolay kullanılır ve sempatik görünümde olabilmesi için, kullanıcının uygulama ile karşı karşıya kalacağı diyalog pencerelerinin özenle tasarlanması gerekir. Kontrol nesneleri, standart kontroller ve dışardan eklenen kontroller olarak iki gruba ayrılabilir. **Standart kontrol nesneleri**, Visual Basic'in yüklenmesi ile hemen kullanılabilecek olan kontrol nesneleridir.

Dışardan eklenen kontroller ise, standart nesnelerin sağladığı imkanların ötesine geçebilen, Microsoft firmasının ya da başka şirketler tarafından üretilmiş ve Visual Basic uygulamasına sonradan herhangi bir zamanda eklenebilecek nesnelerdir. Her iki tip kontrol nesnesinin de müşterek özelliği, daha önceden kendilerine yüklenmiş belirli özelliklere sahip olmalarıdır.

FORM ÜZERİNE KONTROL NESNELERİNİ YERLEŞTİRMEK

Form üzerine kontrol nesneleri yerleştirmek, oldukça kolay olarak gerçekleştirilen bir işlemdir. Aşağıdaki şekilde gerçekleştirilir.

- 1) Form üzerine yerleştirilecek olan kontrol nesnesi, mouse-click (sol tuşu bir kez tıklatma) işlemi ile araç kutusundan (Toolbox) seçilir.
- 2) Mouse yardımı ile, ekran göstergesi form üzerine getirilir. Mouse sol tuşu basılı tutularak mouse hareket ettirilerek kontrol nesnesinin form üzerinde arzu edilen büyüklüğe erişmesi sağlanır. Sonra mouse sol tuşu serbest bırakılır.

2-1-4-1 STANDART KONTROL NESNELERİ VE İŞLEVLERİ



GÖSTERGE

Form üzerine aktarılıp görüntülenemeyen yegane kontrol nesnesidir. Form üzerine çizilmiş bir kontrol nesnesinin boyutunu değiştirmek veya bu kontrol nesnesini hareket ettirmek gerektiği zaman yararlanılır.

**RESİM KUTUSU (PICTURE BOX)**

Resim kutusu nesnesi yardımı ile form üzerinde grafikleri görüntülemek mümkündür. Ayrıca, diskinizin herhangi bir dizini içinden çağıracağınız bir bitmap, ikon(icon) ,bir resimde yükleyebilirsiniz

**ETİKET(LABEL)**

Etiket kontrol nesnesi, kullanıcı tarafından değiştirilemeyecek bir metni görüntülemek ya da icra esnasında uygulama tarafından değiştirilemeyecek bir metni görüntülemek amacı ile kullanılabilir.

**METİN KUTUSU(TEXT BOX)**

Metin kutusu kontrol nesnesi iki ayrı amaç için kullanılabilir
1) Kullanıcı tarafından, uygulamanın icrası esnasında, bilgi girişi amacı ile
2) Uygulama tarafından üretilen bilgilerin ekranda, form üzerinde görüntülenmesi amacı ile
Metin kutusunun Properties penceresinde Multiline özelliği True haline getirilmelidir.

**ÇERÇEVE (FRAME)**

Çerçeve nesnesi, form üzerine yerleştirilecek kontroller için görsel ve fonksiyonel anlamda bir grup oluşturma imkanı sağlar. Bu anlamda, çerçeve içine alınan kontroller, diğerlerinden kolayca ayırdedilirler.

**KOMUT BUTONU(COMMAND BUTTON)**

Komut butonu, kullanıcı tarafından seçildiği zaman, belirli bir işlemi yerine getirir. Bu işlem, kendisine bu olaya bağlı olarak yüklenmiş olan kod ile gerçekleştirilir. Komut butonu mouse yardımı ile bir kez tıklatma(click)işlemi gerçekleştirilebilir. Çeşitli diyalog pencerelerinde karşımıza çıkan OK ve CANCEL butonları komut butonlarına örnektir.

**KONTROL KUTUSU (CHECK BOX)**

Kontrol kutusu, açma ya da kapama gibi ya da seçme ve iptal gibi ikili bir kontrole ihtiyaç duyulan uygulamalarda kullanılır. Kullanıcı , belirtilen seçeneği seçerse, kutu içinde bir işaret sembolü görüntülenir. Kullanıcıya Evet/Hayır gibi seçenekler vermek için kullanılır.

**OPSİYON BUTONU(OPTION BUTTON)**

Option butonları bir grup halinde kullanılırlar. Bu grup içerisinde, her buton bir seçeneği temsil eder. Kullanıcı seçtiği seçeneği, ilgili opsiyon butonunu işaretleyerek belirtir.

**KOMBİNE KUTU (COMBO BOX)**

Bir kombine kutu, metin kutusu(text box) ve liste kutusunun (list box) özelliklerini bir araya toplar. Örneğin, kullanıcı metin kutusu gibi içine metin yazabilir ya da liste kutusu gibi, kutu içine yerleştirilmiş isimlerden istediklerini seçebilir.

**LİSTE KUTUSU (LIST BOX)**

Liste kutusu, kullanıcının içinden bir ya da daha fazla isim seçebileceği bir liste içerir. Listeye yeni isimler eklenebilir ya da listedeki bazı isimler çıkartılabilir. Örneğin, bir liste kutusu ile çeşitli ülke isimlerini listeleyebilirsiniz. Kullanıcı bir isim seçince, o ülke isimlerini listeleyebilirsiniz. Kullanıcı bir isim seçince, o ülkenin özellikleri listelenebilir.

**YATAY KAYDIRMA ÇUBUKLARI(HORIZONTAL SCROLL BARS)**

Kullanıcıya, listeler içinde veya büyük miktarda bilgi boyunca yatay hareket imkanı sağlar. Bu anlamda, bir fonksiyonun değerlerinin görüntülenmesini, bir yatay hareket çubuğunun hareketi ile ilişkili olarak gerçekleştirebilirsiniz.

**DÜŞEY HAREKET ÇUBUKLARI(VERTİCAL SCROLL BARS)**

Düşey hareket çubukları, yatay hareket çubukları için söylenen işlemleri düşey olarak gerçekleştirirler.

**ZAMANLAYICI (TİMER)**

Uygulamanın çalışması esnasında, zamana bağlı olarak belirli aralıklarla, belirli eylemlerin gerçekleşmesini denetlemek amacı ile kullanılır.

**SÜRÜCÜ LİSTELEME KUTUSU**

İcra esnasında, geçerli sürücülerin bir listesini sunmak ve sürücüler arasında geçiş yapmak amacı ile kullanılır . Sürücü listeleme kutusu, bir dosyayı belleğe

yüklemek (açmak-open)amacı ile oluşturulan bir diyalog penceresinin bir parçası olarak kullanılabilir.

**DİZİN LİSTELEME kutusu(DIRECTORY LİST BOX)**

İcra esnasında, aktif durumdaki sürücünün yolunu(path)ve dizinlerini (Directories)görüntüler. Bu kontrol nesnesi, kök dizinden (root directory) seçilmiş bir yolu izleyerek hiyerarşik olarak dizin listesini görüntülemek amacı ile kullanılabilir. Aynı zamanda doğal olarak bir dosya açma diyalog penceresinin bir parçası olarak ta kullanılabilir.

**DOSYA LİSTELEME KUTUSU (FILE LIST BOX)**

Verilen bir dizin içindeki tüm dosyaları listeler. Kullanıcı, listelenmiş dosyalar içinden bir dosya seçebilir. Bu kontrol nesnesi de, bir dosya açma diyalog penceresinin bir parçası olarak kullanılabilir.

**ŞEKİL (SHAPE)**

Tasarım (desing)zamanında görünür olan bir kontrol nesnesidir. Tasarım esnasında, form üzerine , kare,daire,elips,dikdörtgen gibi şekilleri yerleştirmek amacı ile kullanılır.

**ÇİZGİ(LİNE)**

Çizgi kontrol nesnesi, tasarım esnasında, form üzerine yatay, düşey ya da eğik çizgi yerleştirmek amacı ile kullanılır. Çizgi kontrol nesnesi formu parçalara ayırmak amacı ile kullanılabilir.

**GÖRÜNTÜ (IMAGE)**

Bir resmi görüntüleyebilen grafik tipte bir kontrol nesnesidir. Bu açıdan resim kutusuna (picture box) benzer. Fakat, resim kutusuna oranla daha az sistem kaynağı kullanılır. Mouse ile clicklendiği zaman, komut butonuna benzer özellik gösterir. Örneğin bir firma, firmanın logosunu görüntü nesnesi içine yerleştirerek, resim üzerine mouse ile click işlemi gerçek-leştirildiğinde firma hakkında bazı bilgiler listeleyen bir tanıtım yazılımı oluşturabilir.

**VERİYE ERİŞİM (DATA ACCESS)**

Sistemde mevcut olan veri tabanlarına erişerek, bilgi güncelleme, edit ya da bilgi görüntüleme işlemlerinin yapılabilmesine olanak sağlar.

**OLE İSTEMCİSİ (OLE CLIENT)**

Nesneyi bağlama ve yerleştirme yöntemi, Windows işletim sisteminde kullanılan yeni bir teknolojidir. Bu teknoloji sayesinde, bir windows uygulama programı (Mesela Visual Basic 6.0) ile geliştirilen bir proje içine başka bir windows uygulama programından nesneler aktarmak mümkündür.

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 işlemlerle 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.

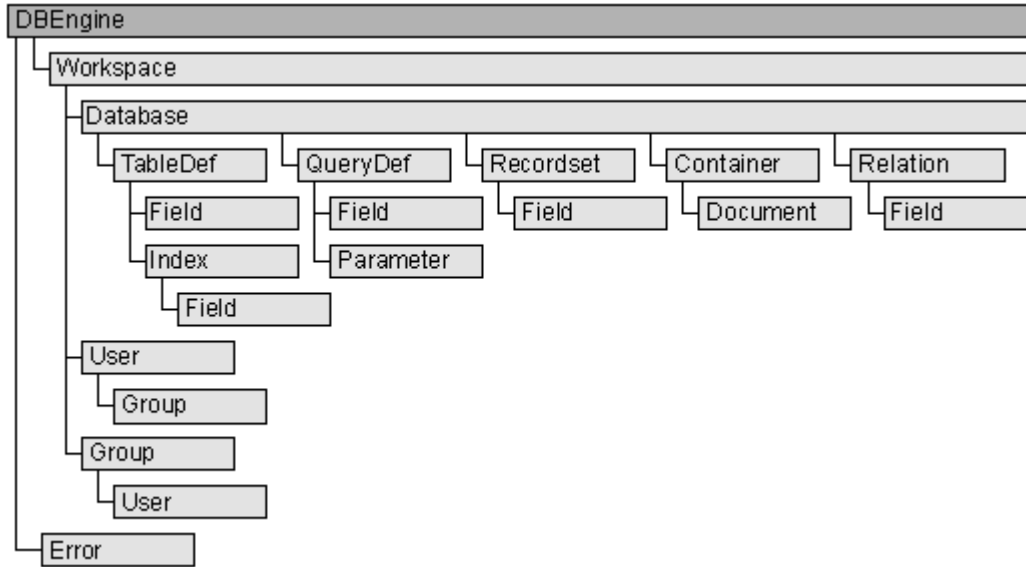
- Uzak veri tabanı sistemi (Remote Database)
- İstemci/Sunucu veritabanı sistemi (Client/Server Database)

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 gruptur.

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 veritabanılarını 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 yanı sıra 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 yanı sıra 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()

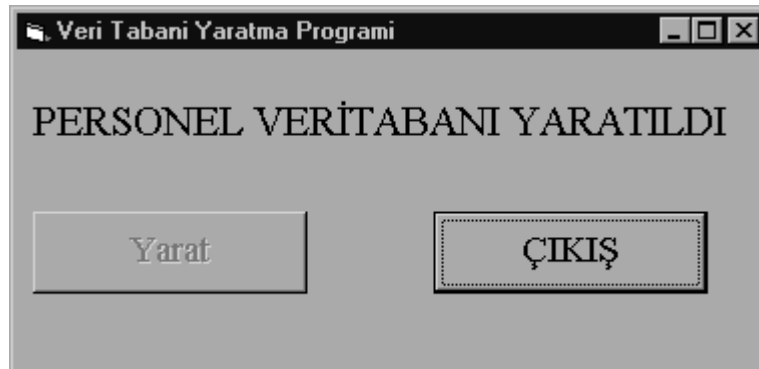
Dim ws As Workspace
Dim db As Database
Dim tdf As TableDef
Dim fld As Field

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 gereken 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 **İndeks 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

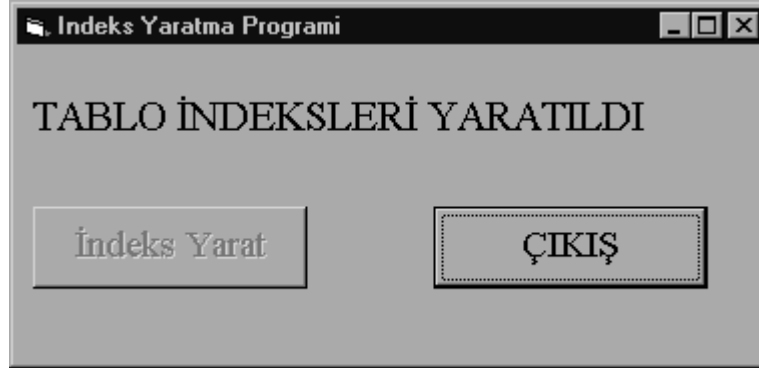
'GOREVLER tablosu için indeks yaratma
Set tdf = db.TableDefs("gorevler")
Set idx = tdf.CreateIndex("ind_gno")
Set fld = idx.CreateField("gno")
idx.Fields.Append fld
Set fld = Nothing
idx.Primary = True
idx.Unique = True
tdf.Indexes.Append idx

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 olarak 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, güncleme 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 yapıp yapılmadığı bilgisini döndüren sadece okunabilir bir özelliktir.

Yukarıdaki kodda görülebileceği gibi veritabanına erişimi destekleyen yordamların yanı sıra hata durumlarını ele alan ve hatalı bir duruma girildiğinde hata veren yordamlar da bulunmaktadır. Bu yordamlar kendi hata mesajlarımızı yazmak ve hatalı durumlara 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  
VeriAI  
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:
```

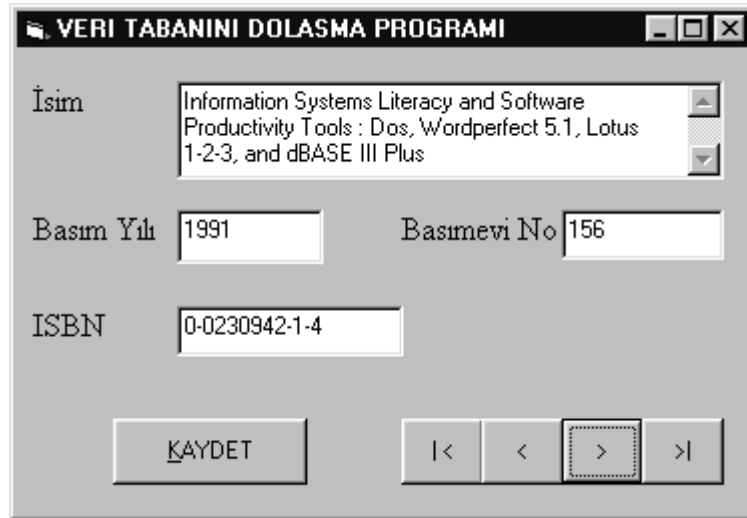


```
MsgBox Err.Description, vbExclamation
Resume pcik
End Sub
Private Sub Text1_Change(Index As Integer)
On Error GoTo phata
If Not okunuyor Then
Select Case Index
Case 0: Kitaplar.ad = Text1(Index).Text
Case 1: Kitaplar.yil = Text1(Index).Text
Case 2: Kitaplar.bno = Text1(Index).Text
Case 3: Kitaplar.ISBN = 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.EOF And rs.BOF 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 = "YANLIŞ 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 metod 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 kullandığımız 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 çıkacaktır.

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	dye
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. İndeks 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 parametrelerini 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 çağırılıyor. 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.

7.İLERİ DÜZEY VERİ TABANI GELİŞTİRME

7.1. VERİ BÜTÜNLÜĞÜNÜ SAĞLAMA.

Visual Basic ile veritabanı geliştirirken sağlanan önemli bir özellik de veri bütünlüğünü koruma özelliğidir. Bu ilişkisel veritabanı kullanımının temel bir gereğidir.

Visual Basic ile veritabanının veri bütünlüğünü korumak ve sağlamak için veritabanı tabloları arasında tanımlanabilen ilişkiler kullanılır. İlişki tanımlama kavramına daha önce değinilmişti. İlişki yukarıda da bahsedildiği gibi birbiriyle ilişkili alanlar barındıran iki farklı tabloya girilecek verilerin tutarlılığını sağlar. Veritabanı üzerinde bir ilişki tanımlandıktan sonra, ilişkiyi ihlal eden veri girişi DAO nesnesinin ele alınabilir hatalı bir duruma girmesini sağlar. Böylece hem veri bütünlüğü ve tutarlılığı sağlanmış olur, hem de kullanıcı böyle bir ihlalin açığa çıktığından haberdar edilir.

Veritabanı bütünlüğünü sağlamanın bir diğer yolu da bazı kurallar tanımlayarak kullanıcının belirli alanlara yanlış veriler girmesine engel olmaktır.

Veritabanı üzerinde tablolara ve alanlara uygulanabilecek birçok kural vardır.

- Alanlara girilebilecek değerler bir liste ile sınırlanabilir.
- Alanlara veri girilmesi zorunlu hale getirilebilir.
- Alanlara girilen değerler başka alanlara girilen değerlere bağlı olabilir.

Kuralları uygulamanın iki yolu vardır.

- Ekleme, silme ve değiştirme yapmadan önce verilerin kurallara uyup uymadığını kontrol etmek.
- Veritabanı tablolarına kuralları ekleyerek veritabanı motorunun kuralları uygulamasını sağlamak.

Veritabanı üzerinde tanımlanabilecek kuralların hepsini veritabanı motoruna bırakmak pek mümkün değildir, fakat amaç olabildiğince çok kuralı veritabanı tanımına eklemek ve kuralları veritabanı motorunun uygulamasını sağlamak olmalıdır.

Şu ana kadar veritabanı üzerinde bütünlüğünü sağlamaya yönelik bir çok kural tanımladık bunlar;

- Bir alanın veri türünü belirleme (Yanlış türden veri girişini engeller),
 - Primary Key ve Unique indeks tanımlama (verilerin tekrarını engeller),
 - Tablolar arasında ilişki tanımlama (tutarsız veri girişini engeller)
- kurallarıdır.

Bu kuralların yanısıra aşağıdaki kuralları da veritabanına eklemek mümkündür.

- **Required** özelliği bir alanın boş bırakılarak geçilmesini engeller.
- Bir alanın veya bir tablonun **ValidationRule** özelliği alana veya tabloya girilecek olan değerleri belirli bir koşulla sınırlar. Bu özellik kullanıldığında **ValidationText** özelliği de kullanılır. Bu özellik de bu kural ihlal edildiğinde verilecek hata mesajını belirtir.
- **AllowZeroLength** özelliği metin türünde tanımlanmış bir alanın boş bir değer alabilmesine izin verilmesini sağlar.

Yukarıda belirtilen özelliklerden **Required** ve **AllowZeroLength** özellikleri Boolean türünden özelliklerdir.

ValidationRule özelliği geçerli bir Visual Basic ifadesini belirtir. Bu ifadede kullanıcının tanımladığı fonksiyonlar, SQL kümeleme fonksiyonları, bir sorgu, başka bir tablonun alanı yer **alamaz**.

Şimdi veritabanı bütünlüğünü korumak için kural tanımlarını örneklemek için daha önce kullanmış olduğumuz PERSONEL veritabanı üzerinde bazı kuralları tanımlayalım ve bunları veritabanı üzerine uygulayalım.

1- Önce kuralları bir tablo halinde gösterelim.

Tablolar	Alanlar	Özellik	Değer
Personel_bil	sicno	Required	True
	ad_soyad	Required	True
	dtar	Required	True
	dtar	ValidationRule	Koşul1*
	dyer	Required	True
personel_gorev	sicno	Required	True
	CYIL	Required	True
	CYIL	ValidationRule	Koşul2*
	gno	Required	True
	mno	Required	True
gorevler	derece	Required	True
	gno	Required	True
	GADI	Required	True
meslekler	mno	Required	True
	MADI	Required	True

Koşul1: If(Year(dtar)>1930, Year(dtar)<1976, False)

Koşul2: If(CYIL>=0, CYIL<68, False)

2- Yeni bir proje yaratın. Aşağıdaki nesneleri ekleyin.

Form	Name	frmKurallar
	Caption	KURAL EKLEME PROGRAMI
	BorderStyle	3-FixedDialog
	CommonDialog Name	dlg
CommandButton	Name	cmdKural
	Caption	&Kural Ekle
CommandButton	Name	cmdVT
	Caption	&Veritabanı aç
Label	Name	lblSonuc
	Autosize	True

Forma aşağıdaki kodu girin.

```
Private db As Database
Private Sub cmdKural_Click()
On Error GoTo phata:
KuralEkle
lblSonuc.Caption = "KURALLAR EKLENDİ"
Exit Sub
phata:
MsgBox Err.Description, vbExclamation
End Sub
```

```
Private Sub cmdVT_Click()

Dim dosya As String

dlg.InitDir = App.Path

dlg.DefaultExt = ".mdb"

dlg.DialogTitle = "Veri Tabani Aç"

dlg.Filter = "VB Veritabani (*.mdb)|*.mdb"

dlg.FilterIndex = 1

dlg.Flags = cdIOFNHideReadOnly Or cdIOFNFileMustExist _
Or cdIOFNPathMustExist

dlg.CancelError = True

dlg.ShowOpen

dosya = dlg.filename

Set db = DBEngine.Workspaces(0).OpenDatabase(dosya)

cmdKural.Enabled = True

cmdVT.Enabled = False

Exit Sub

phata:

dosya = ""

End Sub

Private Sub KuralEkle()

Dim td As TableDef

Dim fld As Field

For Each fld In db.TableDefs("personel_bil").Fields
```

```
fld.Required = True

Next

For Each fld In db.TableDefs("personel_gorev").Fields

fld.Required = True

Next

For Each fld In db.TableDefs("gorevler").Fields

fld.Required = True

Next

For Each fld In db.TableDefs("meslekler").Fields

fld.Required = True

Next

Set fld = db.TableDefs("personel_bil").Fields("dtar")

fld.ValidationRule = "IIf(Year(dtar)>1930, Year(dtar)<1976, False)"

fld.ValidationText = "Dogum tarihi 1930-1976 arasinda olmalı"

Set fld = Nothing

Set fld = db.TableDefs("personel_gorev").Fields("CYIL")

fld.ValidationRule = "IIf(CYIL>=0, CYIL<68, False)"

fld.ValidationText = "Çalıştığı yıl 0-68 arası olmalı"

Set fld = Nothing

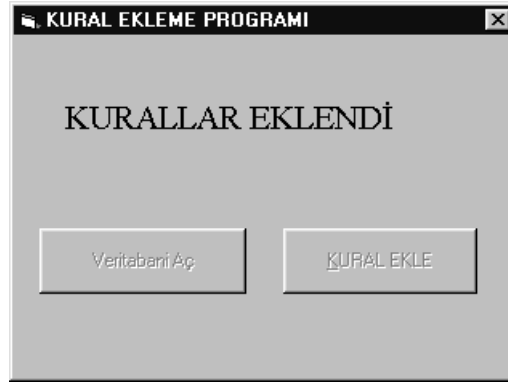
cmdKural.Enabled = False

End Sub
```

Yukarıdaki kodda önemli olan KuralEkle fonksiyonudur. Burada, kural tanımlarımızın gereği tabloların bütün alanlarının **Required** özellikleri **True** yapılmıştır. Bunun anlamı, 'veritabanının hiç bir tablosundaki hiç bir alan boş geçilemez' dir.

Bu kuraldan başka iki tane de farklı kural tanımlanmıştır. Bu kurallar doğum tarihinin yılını ve kişinin kurumda çalıştığı yıl sayısını sınırlayan kurallardır.

Program çalıştığında aşağıdaki form çıkar. Başlangıçta bu formun üstündeki "KURALLAR EKLENDİ" yazısı yoktur ve sadece veritabanı aç butonu aktif haldedir. Bu butona basıldığı anda standart Windows95 Open Dialog kutusu çıkar ve buradan veritabanının adının bulunmasına olanak tanınır. Daha sonra veri tabanı açılır, veritabanı açma butonu gri hale getirilir, kural ekleme butonu aktif hale gelir. Kurallar eklendikten sonra bu buton da gri hale gelir.



7.2. ÇOK KULLANICILI VERİTABANLARI VE GÜVENLİK

Eğer bir ağ üzerinde, birden fazla kullanıcıya hizmet veren uygulamalar geliştirilecekse, bu uygulamanın verileri kullanıcılar arasında etkin bir biçimde paylaşılması gerekir. Bu bölümde, çok kullanıcıli uygulamalar geliştirilirken dikkat edilmesi gereken konuları içermektedir. Çok kullanıcıli sistemlerde verilere erişimin denetlenmesi için gerekli olan kilitleme stratejileri ve bu kilitlemelerin çok kullanıcıli bir ortamda ele alınışına değinilecektir.

Bu bölümde Microsoft Jet veritabanı motoru bağlamında çok kullanıcıli veritabanı kavramı ele alınacaktır, ODBCDirect bağlamında değil. ODBCDirect yöntemi ile uygulamalar doğrudan ODBC veritabanılarına erişebilir. Bu erişim için Jet veritabanı motoruna ihtiyaç yoktur. Bu konuya daha sonra değinilecektir.

Bir programın çok kullanıcıli olabilmesi için;

- 1- Uygulamanın tamamını bir ağ sunucusu üzerine yerleştirmek

2- Uygulamayı iki parçaya bölmek (geri uç kısmı, tabloları içeren kısım ve ön uç kısmı, nesneleri, makroları ve modülleri içeren kısım) ve daha sonra geri uç kısmını bir ağ sunucusuna yerleştirmek ve ön uç kısmını da kullanıcılara dağıtmak

olmak üzere iki yaklaşım sözkonusudur.

Bu yaklaşımlardan ilki ağ trafiğinde bir artışa neden olur.

Ağ üzerinden erişim daha yavaş olduğu için uygulamanın bazı kısımlarını kullanıcı bilgisayarında tutmak her zaman için hız avantajı sağlar. Bu bağlamda uygulamanın kesimleri iki kategori altında toplanabilir.

Statik kesimler : Programın çalışmasını sağlayan .dll ve .exe dosyalarıdır. Uygulamadaki, formlar, raporlar, program kodları da statik kesimlerdir.

Aktif kesimler : Çok kullanıcıli bir ortamda kullanıcıların erişeceği gerçek verilerin tutulduğu dosyalardır. Bunlar Jet veritabanları veya dış veritabanları olabilir.

Statik kesimlerin özellikleri:

- Kullanıcı bilgisayarında kurulu ve çalışır halde olmalıdırlar.
- Uygulamanın tasarımı değiştiğinde bu kısımlar da değişmelidir.
- Uygulamanın nesne yapısındaki değişiklikler anında kullanıcı bilgisayarına yansımak zorundadır. Jet veritabanı motoru bu özelliği sağlayacak bir kopyalama (replication) mekanizmasında sahiptir. Gerekiyorsa bu mekanizma da nesnelerin son biçimlerini elde etmek için kullanılabilir.

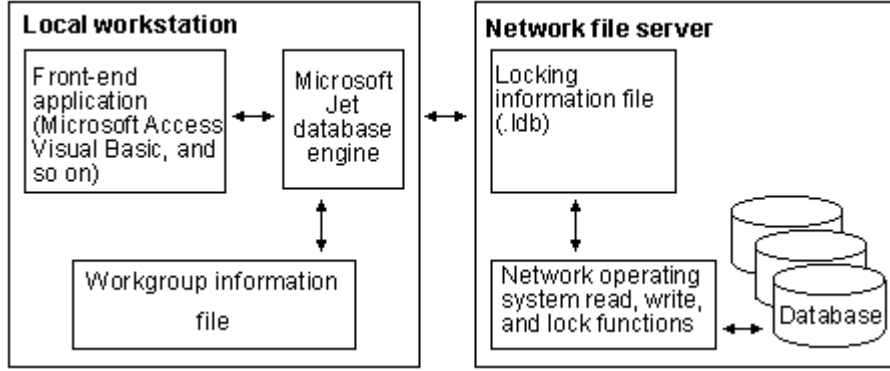
Aktif kesimlerin özellikleri:

- Veri erişimi etkin olmalı, indeks yapıları kullanılmalıdır. Aksi takdirde ağ trafiği artacağından uygulamaların performansı düşer.
- Sık değişmeyen verilerin kullanıcı bilgisayarlarında tutulması ağ trafiğini azaltacaktır.

İki düzeyli veritabanı yaklaşımı sözkonusu olduğunda, her kullanıcının kendi bilgisayarında;

- Bir wokgroup bilgi dosyası ve
- Her bir veritabanı için bir kilitleme bilgi dosyası(.ldb) olması gerekir.

Microsoft Jet veritabanının çok kullanıcıli modeli ařağıdaki gibidir.



Workgroup bilgi dosyası, kullanıcı, grup ve password bilgilerini ve kullanıcı ile ilgili başka bilgileri tutan bir veritabanıdır. Bu dosya ağı sunucusunda veya her bir kullanıcı bilgisayarında tutulabilir. Herbir kullanıcı bilgisayarında tutulduğunda güvenlik bilgileri değiştiğinde değişiklikleri anında güncelleme gerekliliğı vardır.

Kilitleme bilgi dosyası, kilitlenmiş olan tutanaklarla ilgili bilgileri tutar. Veri tabanı açıldığında bu kütük yoksa jet veritabanı motoru bu dosyayı yaratır. Paylaşımlı olarak açılan herbir veritabanı dosyası için bir kilitleme dosyası tutulur. Bu dosyaya verilen isim veritabanı dosyasının ismiyle aynıdır. Bu dosya veritabanı dosyası ile aynı dizinde tutulur.

Microsoft Jet tarafından desteklenen ağı-sunucuları řunlardır:

Microsoft Windows NT and Windows 95 networks

Banyan VINES 5.52

DECnet 4.1

LANtastic 5.0

Novell NetWare 3.x and 4.x

OS/2 LAN Manager, 2.1 and 2.2

Tek kullanıcıli bir ortamda çalışırken veriler tamamen kullanıcıya ait olduğundan hiç bir sorun yoktur. Fakat çok kullanıcıli ortama geçildiğinde çatışmalar söz konusudur.

Bu çatışmalar kitleme olanakları sayesinde engellenmektedir. Bu olanaklar birçok kullanıcının aynı anda veritabanı üzerinde işlem yapmasına olanak tanır.

Paylaşımlı bir veritabanı üzerinde işlem yapılırken, değişiklik yapılacağı zaman veri kilitlenir. Bu sırada başka kullanıcılar bu veriyi okuyabilir fakat değişiklikler yapamaz. Jet veritabanı motoru üç düzeyde kitleme olanağı sağlamaktadır.

Dışlayan mod : Tüm kullanıcıların veritabanına erişimlerini engeller.

Bu en kısıtlayıcı moddur.

RecordSet kilitleme : Bir Recordset nesnesinin belirttiği bir tabloyu table-read veya table-write kilitlerini kullanarak kilitler.

Sayfa kilitleme : Üzerinde değişiklik yapılmakta olan 2048 byte(2K) uzunluğundaki veriyi kilitler.

Hangi tür kilitlemeyi kullanacağımızı belirlemek için, sağlanacak olan paralellik düzeyi dikkate alınmalıdır. En yüksek paralellik için en düşük kısıtlamaya sahip kilitlenme kullanılmalıdır. Bunun yanı sıra tüm verilere çabuk bir şekilde erişim için ise dışlayan kilitleme tercih edilmelidir.

Veritabanı üzerinde büyük değişiklikler yapılacaksa dışlayan mod tercih edilmelidir.

Tek kullanıcı bir ortamda veritabanı erişimleri dışlayan modda olur, çünkü veritabanını kullanacak başka kimse zaten yoktur. Çok kullanıcı sistemlerde veritabanı paylaşımlı modda kullanılmak üzere düzenlenmelidir. Bu tür ortamlarda kullanıcıların veritabanını dışlayan modda açmaları engellenmelidir. Bunun için kullanıcılara veritabanı üzerinde haklar tanımlanır. Bir kullanıcı bu haklar çerçevesinde veritabanına erişir.

Tutanak bazında kilitleme söz konusu ise, veritabanı paylaşımlı modda açılır. Veritabanı paylaşımlı modda açıldığında Jet veritabanı motoru çatışmaları ele alır ve tutarsız durumlara girilmesini engeller.

Aşağıdaki kod parçası, veritabanının paylaşımlı modda açılmasını örneklemektedir.

```
Function VTac(vt As Database, dosya As String, dislayan As Boolean) As Integer
On Error Resume Next

Set vt = OpenDatabase(dosya, dislayan)

Select Case Err

Case 0: VTac = 0 'Başarılı
```

```
Case 3033: VTac = 1 'Erişim Hakkı yok
Case 3343: VTac = 2 'Hatalı Veritabanı
Case 3044: VTac = 3 'Hatalı Dizin Adı
Case 3024: VTac = 4 'Hatalı Dosya Adı
Case Else: VTac = 5 'Başarısız

End Select

Err = 0

End Function
```

Bu kod sayesinde veritabanı verilen parametreye göre paylaşımlı veya paylaşımsız açılmakta ve bu açma işleminin sonucu çağrıldığı ortama 0-5 arasında bir sayı olarak döndürülmektedir. Bu fonksiyonun kullanımı şu şekilde olabilir.

```
Dim Sonuc As Integer

Dim db as Database

Sonuc = VTac(db,"PERSONEL.MDB", False)
```

Bazen bir veritabanını sadece okunabilir modda açmak gerekebilir. Bu mod paylaşımlı modun değiştirilmiş biçimidir. Bir veritabanını sadece okunabilir modda açmak için şu kod kullanılır.

Vt = OpenDatabase("PERSONEL.MDB", False, True)

Veritabanı sadece okunabilir modda açıldığı zaman hiç bir çatışmaya neden olunmadan veritabanı dolaşılabilir.

Recordset kilitleme yöntemiyle, Recordset nesnesinin ilgili olduğu tutanakların kilitlenmesi sağlanabilir. Bu tür kilitleme ancak Dynaset türü Recordset nesnelerinde mümkündür. Veritabanı paylaşımlı modda açıldıktan sonra Recordset kilitlemeyi sağlamak için OpenRecordset metodu birkaç parametre daha eklenerek çağrılır.

Kilitlemeyi sağlamak için,

- 1- Veritabanı paylaşımlı modda açılır,
- 2- OpenRecordset metodu, kilitleme biçimi parametre olarak verilerek çağrılır.
- 3- Recordset nesnesi ile yapılacak işlemler bittiğinde Close metodu çağrılarak kilitlemenin iptal edilmesi sağlanır.

Aşağıdaki kod kesiminde bir Recordset Yapısının dışlayan modda nasıl açılabileceğini görmek mümkündür.

```
Function TabloAc(vt As Database, rs As Recordset, Tablo As String) As Integer
Set rst = dbs.OpenRecordset(Tablo, _
dbOpenTable, dbDenyRead + dbDenyWrite)
Select Case Err
Case 0: OpenTableExclusive = 0 'Başarılı
Case Else: OpenTableExclusive = 1 'Başarısız
End Select
Err = 0
End Function
```

Burada Recorset yapısı verilen 3. Parametre ile hem yazmaya hem okumaya karşı kilitlemiştir. Bu parametre belirtilmezse Sayfa Kilitleme yöntemi kullanılır. Bu yöntemle başka kullanıcıların veriyi okumaları engellenmez.

Recordset açılırken eğer başka kullanıcılar tarafından kullanılan tutanaklara kilitleme yapılmaya çalışılıyorsa aşağıdaki hata oluşur.

```
error 3262:"Couldn't lock table <a>; currently in use by user <b> on machine <c>."
```


Bu hatada a, bir tablo ismi, b, bir kullanıcı ismi, c, ağdaki bir makinenin ismidir. Bu hatayı ele almak için bir süre bekliyerek kilitlemeyi tekrar denemek gerekir.

Sayfa Kilitleme yöntemiyle, değiştirilecek tutanakların kilitlenmesi mümkündür. Bir seferde 2048 byte uzunluğundaki veri bu yöntemle kilitlenebilir. Bu yöntemle veriler kilitlendiğinde diğer kullanıcılar veriyi okuyabilir fakat değiştiremez. Sayfa kilitlemeyi denetlemek için Recordset nesnesinin **LockEdits** özelliği kullanılır. Sayfa kilitleme modları iki tanedir. Bunlar **Pessimistic** ve **Optimistic** olarak adlandırılır.

Pessimistic Kilitleme ile, şu anda değiştirilecek olan tutanağın bulunduğu sayfa, **Edit** metodu kullanıldığı anda kilitlenir. Bu kilitleme değişiklikler tamamlanıncaya kadar devam eder.

Bu kilitleme yönteminin temel avantajı, kilitleme işlemi yapıldıktan sonra kilitlenen sayfadaki tutanaklarla yapılan işlemlerde çatışma olmayacağını garanti edilmesidir. Ayrıca, pessimistic kilitleme veritabanındaki tutanakların en son durumunu elde etmenin en emin yoludur. Çünkü bir kere kilitleme yapıldıktan sonra hiç bir kullanıcı bu verileri değiştiremez.

Dezavantajı ise, sayfanın uzun süre kilitli kalma riskidir. Örneğin bir kullanıcı sayfayı kilitler ve uzun süre bilgisayarı boş bırakırsa bu sayfada uzun bir süre boyunca kimse değişiklik yapamaz.

Optimistic Kilitleme yöntemiyle, sayfa sadece değişiklikler kaydedileceği zaman kilitleme yapılır. Böylece sayfanın kilitli kaldığı süre önemli ölçüde azalmış olur.

Bu kilitleme yönteminin dezavantajı da, yapılan değişikliklerin kaydedilebileceğinden emin olamamaktır. Çünkü değişikliğe başlandığında gerçek bir kilitleme yapılmaz. Bu esnada başka bir kullanıcı tutanağı pessimistic kilitleme yöntemiyle kilitlerse, yapılan değişiklikler kaydedilmeye çalışıldığında erişim hakkı olmadığı için bu işlem başarısız olacaktır.

Optimistic Kilitlemeyi kullanmak için;

- 1- Tablo veya Dynaset türünde bir Recordset açın.
- 2- Belirli bir tutanağı aktif tutanak yapın.
- 3- Recordset nesnesinin LockEdits özelliğine False aktarın.
- 4- Edit metodunu kullanarak değişikliğe başlayın.
- 5- Update metodunu kullanarak değişiklikleri kaydedin

6- Başarısızlık durumunda tekrar deneyin.

Değişiklikler kaydedileceği zaman veritabanı üzerinde Pesimistic kilitleme otomatik olarak yapılır.

Sayfa Kilitleme sırasında çıkan hatalar:

3188. Daha önce kilitlenmiş sayfa, kaydetme yapılamaz.

3197. Aynı anda değişiklik yapmaya kalkışma sonucu oluşan çatışma.

3260. Daha önce kilitlenmiş sayfa, kilitleme yapılamaz.

Örnek :

Sayfa kilitlemeyi örneklemek için daha önce kullanılan PERSONEL veritabanının PERSONEL_BIL tablosunu kullanalım.

1- Yeni bir proje yaratın ve içindeki formu silin. Projeye frmPBil formunu ekleyin.

2- Formun kodunu tamamen silin.

3- Formun üzerindeki, SIL butonunu YENILE, EKLE butonunu DEGISTIR olarak değiştirin. İsimlerini de buna uygun hale getirin.

4- Form üzerindeki BUL butonunu silip yerine aşağıdaki nesneleri ekleyin.

Frame Name Frame1

OptionButton Name Option1

Caption Pessimistic Kilitleme

OptionButton Name Option2

Caption Optimistic Kilitleme

5- Formun kod kısmına aşağıdaki kodu ekleyin.

```
Option Explicit

Private db As Database

Private rs As Recordset

Private degisiyor As Boolean

Private Const PESSIMISTIC = True

Private Const OPTIMISTIC = False

Private Sub cmdDeg_Click()

Dim i As Integer

On Error GoTo phata

If degisiyor Then TutanakGunle

rs.Edit

degisiyor = True

cmdDeg.Enabled = False

cmdKaydet.Enabled = True

For i = 0 To 3

Text1(i).Enabled = True

Next i

Exit Sub

phata:

Select Case Err.Number

Case 3260: MsgBox "Tutanak Kilitli", vbExclamation, "degistirme"
```

```
Case Else: MsgBox Err.Description, vbExclamation, "degistirme"

End Select

End Sub

Private Sub cmdKaydet_Click()

TutanakGunle

End Sub

Private Sub cmdYenile_Click()

If degisiyor Then TutanakGunle

rs.Requery

rs.MoveNext

rs.MovePrevious

VeriAl

End Sub

Private Sub Form_Load()

Dim dosya As String

On Error GoTo phata

dosya = App.Path & "\PERSONEL.MDB"

Set db = DBEngine.Workspaces(0).OpenDatabase(dosya)

Set rs = db.OpenRecordset("personel_bil", dbOpenDynaset)

If rs.EOF And rs.BOF Then

MsgBox "Bos Tablo", vbExclamation, "HATA"
```



```
Unload Me

Else

rs.MoveFirst

VeriAl

End If

Option1 = True

pcik: Exit Sub

phata:

MsgBox Err.Description, vbExclamation

Unload Me

Resume pcik

End Sub

Private Sub Form_Unload(Cancel As Integer)

Set db = Nothing

Set rs = Nothing

End Sub

Private Sub Command1_Click(Index As Integer)

On Error GoTo phata

If degisiyor Then TutanakGunle

Select Case Index

Case 0: rs.MoveFirst

Case 1:

rs.MovePrevious
```

```
If rs.EOF Then rs.MoveFirst

Case 2:

rs.MoveNext

If rs.EOF Then rs.MoveLast

Case 3: rs.MoveLast

End Select

VeriAl

pcik: Exit Sub

phata:

MsgBox Err.Description, vbExclamation

Resume pcik

End Sub

Private Sub Option1_Click()

If degisiyor Then TutanakGunle

rs.LockEdits = PESSIMISTIC

End Sub

Private Sub Option2_Click()

If degisiyor Then TutanakGunle

rs.LockEdits = OPTIMISTIC

End Sub

Private Sub VeriAl()

Dim i As Integer

Text1(0).Text = rs!sicno
```

```
Text1(1).Text = rs!ad_soyad

Text1(2).Text = rs!dyer

Text1(3).Text = rs!dtar

For i = 0 To 3

Text1(i).Enabled = False

Next i

cmdDeg.Enabled = True

cmdKaydet.Enabled = False

degisiyor = False

End Sub

Private Sub TutanakGunle()

On Error GoTo phata:

rs!sicno = Text1(0)

rs!ad_soyad = Text1(1)

rs!dyer = Text1(2)

rs!dtar = Text1(3)

VeriAl

Exit Sub

phata:

Select Case Err.Number

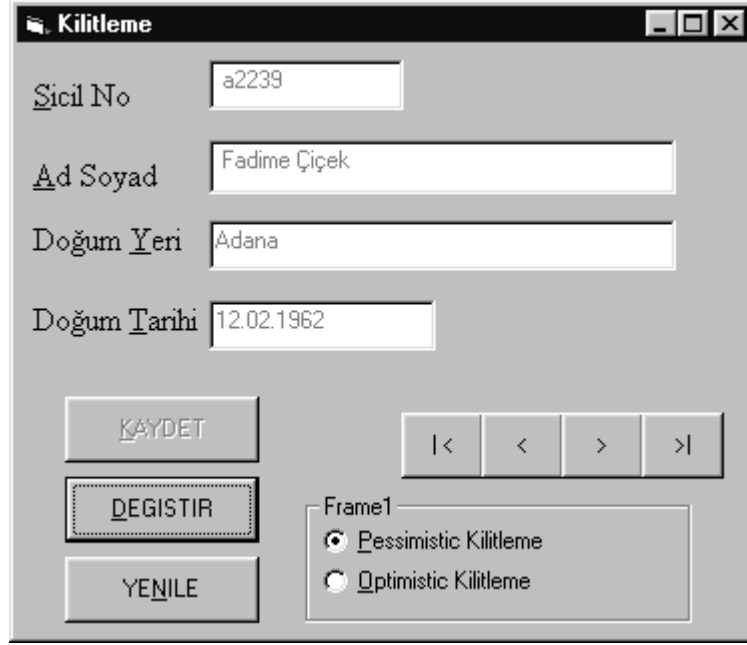
Case 3260: MsgBox "Tutanak Kilitli", vbExclamation, "Kaydetme"

Case Else: MsgBox Err.Description, vbExclamation, "Kaydetme"

End Select

End Sub
```

Programın çalışması aşağıdaki gibi olacaktır.



Kodu açıklayacak olursak; form yüklenirken veritabanı Recordset yapısı açılıyor. Değiştir butonuna basıldığında veritabanı üzerinde değişiklik yapma başlatılıyor. Bu durumda tutanak kilitleme önem kazanmaktadır.

Yenile butonuyla aktif tutanak yeniden gösteriliyor. Böylece son yapılan değişikliklerin görülmesi sağlanabilir.

VeriAl yordamında veriler metin kutularına alınıyor. TutanakGunle yordamında ise değiştirilmiş olan veriler veritabanı üzerine kaydediliyor.

Şimdi yukarıda anlatılan kilitleme mekanizmasını denemek için bir program hazırlanmış oldu. Bu program kullanılarak optimistic ve pessimistic kilitleme arasındaki fark görülebilir.

Bu programın başka kopyalarını çalıştırarak çok kullanıcı ortam benzetimi yapılabilir.

7.3. DIŞ (EXTERNAL) VERİTABANLARINA ERİŞİM

Microsoft Jet veritabanı motoru Microsoft Access veritabanları üzerinde çalışır. Bunun örneklerini şimdiye kadar gördük. Bu veritabanlarından başka bir çok veritabanının da Jet veritabanı motoru tarafından desteklendiği daha önce belirtilmişti. Bu bölümde bunun gerçekleştirimi örneklenecektir.

Microsoft Access veritabanı olmayan tüm veritabanları burada Dış(External) veritabanı olarak adlandırılmaktadır. Bu veritabanları üzerinde işlem yapılırken bazı

kısıtlamalar olmasına rağmen yine de bir çok kullanışlı metot ve özellik sunulmaktadır.

Dış veritabanları iki sınıfta toplanır. Bunlar ISAM ve ODBC veritabanlarıdır.

ISAM veritabanlarına erişim için üç yol vardır.

- 1-** Bound Control ve Data Control nesnelerini kullanmak (bunun Access veritabanlarına erişmekten farkı yoktur, sadece Connect özelliğini belirtmek yeterlidir).
- 2-** DAO kullanmak
- 3-** Bir tabloyu Access veritabanına bağlayıp Recordset yapısıyla tabloya erişmek.

Örnek. DAO kullanarak Microsoft Excel veritabanlarına erişim.

Excel veritabanlarına DAO kullanarak erişim yapılırken bazı kısıtlamalar söz konusudur. Bunlar;

- Satırlar silinemez
- Formül içeren hücreler silinemez ve değiştirilemez
- İndeks yaratılamaz
- Şifrelenmiş excel veritabanları kullanılamaz.

Bunların dışında Excel veritabanları tıpkı Access veritabanıymış gibi kullanılır. (Dış veritabanları için var olan bazı temel kısıtlamalar dışında)

Örnek :

- 1-** Yeni bir proje yaratın ve ismini ExcelDAO olarak değiştirin. Bu proje için PERSONEL veritabanının PERSONEL_BIL tablosunun excelde yaratılmış hali kullanılacaktır.
- 2-** Aşağıdaki nesneleri bir formun üzerine ekleyin ve özelliklerini belirtildiği gibi değiştirin.

Form	Name	frmPBil
	Caption	EXCEL VERİTABANI
ListView	Name	ListView1
	View	3-lvwReport
CommandButton	Name	cmdEkle
	Caption	&Ekle
CommandButton	Name	cmdDeg
	Caption	&Değiştir
CommandButton	Name	cmdYenile
	Caption	&Yenile
CommandButton	Name	cmdBak
	Caption	E&xcel'de Bak
CommandButton	Name	cmdKapat
	Caption	&Kapat
CommandButton	Name	cmdTamam
	Caption	&Tamam
CommandButton	Name	cmdIptal
	Caption	İ&ptal
Textbox	Name	Text1 (Control array)

Formun tasarım esnasındaki görünüşü aşağıdaki gibi olmalıdır.



Aşağıdaki kodu formun kod kesimine yazın.

```
Option Explicit

Private db As Database
Private rs As Recordset
Private durum As Integer
Private Const EKLEME = 0
Private Const DEGISTIRME = 1
Private aktifHucre As ComctlLib.ListItem

Private Sub cmdBak_Click()

Set rs = Nothing

Set db = Nothing

Shell "d:\msoffice\office\excel.exe " & App.Path & "\personel.xls", _
vbNormalFocus

End Sub

Private Sub cmdDeg_Click()

Dim i As Integer

rs.MoveFirst

For i = 1 To ListView1.SelectedItem.Index - 1

rs.MoveNext

Next i

Text1(0) = rs!sicno
```

```
Text1(1) = rs!ad_soyad
Text1(2) = rs!dtar
Text1(3) = rs!dyer
EkleGoster
durum = DEGISTIRME
End Sub

Private Sub cmdEkle_Click()
Dim i As Integer
For i = 0 To 3
Text1(i) = ""
Next
EkleGoster
durum = EKLEME
End Sub

Private Sub cmdIptal_Click()
EkleSakla
End Sub

Private Sub cmdKapat_Click()
,Unload Me
End Sub

Private Sub cmdTamam_Click()
If durum = EKLEME Then rs.AddNew Else rs.Edit
rs!sicno = Text1(0)
```

```
rs!ad_soyad = Text1(1)
rs!dtar = Text1(2)
rs!dyer = Text1(3)
rs.Update
ListeDoldur
EkleSakla
End Sub

Private Sub cmdYenile_Click()
ListeDoldur
End Sub

Private Sub Form_Activate()
DoEvents
ListeDoldur
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set db = Nothing
Set rs = Nothing
End Sub

Private Sub EkleSakla()
Me.Height = 3700
NesneGunle True
End Sub

Private Sub EkleGoster()
```



```
Me.Height = 5265

NesneGunle False

End Sub

Private Sub NesneGunle(idurum As Boolean)

Dim i As Integer

cmdEkle.Enabled = idurum

cmdDeg.Enabled = idurum

cmdYenile.Enabled = idurum

cmdBak.Enabled = idurum

cmdKapat.Enabled = idurum

For i = 0 To 3

Text1(i).Enabled = Not idurum

Next

cmdTamam.Enabled = Not idurum

cmdIptal.Enabled = Not idurum

End Sub

Private Sub ListeDoldur()

Dim fld As Field

Dim alanSay As Integer

Dim alanGen As Single

Dim listeEl As ListItem

EkleSakla
```



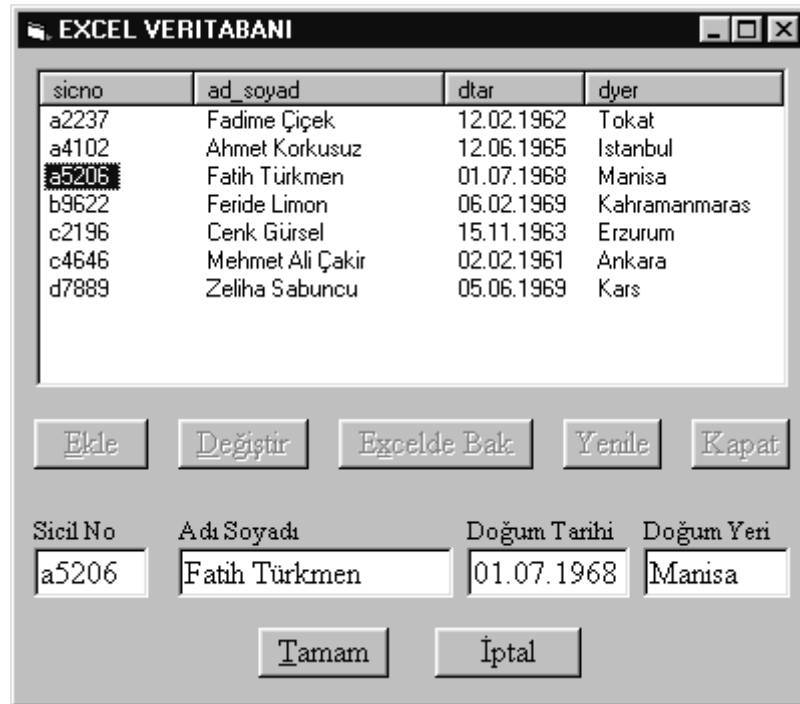
```
Set db = OpenDatabase(App.Path & "\personel.xls", False, False, _  
"Excel 8.0;HDR=YES;")  
Set rs = db.OpenRecordset("personel_bil$")  
With ListView1  
    .ColumnHeaders.Clear  
    .ListItems.Clear  
    For Each fld In db.TableDefs("personel_bil$").Fields  
        alanGen = TextWidth(fld.Name) + 500  
        .ColumnHeaders.Add , , fld.Name, alanGen, vbLeftJustify  
    Next fld  
    .ColumnHeaders.Item(4).Width = .ColumnHeaders.Item(4).Width + 400  
    .ColumnHeaders.Item(2).Width = .ColumnHeaders.Item(2).Width + 400  
End With  
rs.MoveFirst  
While (Not rs.EOF)  
    Set listeEl = ListView1.ListItems.Add(, , CStr(rs.Fields(0)))  
    For alanSay = 1 To rs.Fields.Count - 1  
        listeEl.SubItems(alanSay) = rs.Fields(alanSay)  
    Next alanSay  
    rs.MoveNext  
Wend
```

```
Set aktifHucre = listeEl
```

```
Set listeEl = Nothing
```

```
End Sub
```

Program biraz incelendiğinde görülecektir ki erişimin daha önceki veritabanı erişimlerden hiç farkı yoktur. Tek fark veritabanını açma aşamasındadır. Excel için bir diğer fark da Tablo isimlerinin sonuna “\$” işaretinin konması gerekliliğidir. Programın çalışması aşağıdaki gibi olacaktır.



Excel Veritabanı formu, bir tablo ve form alanları içerir. Tablo, sicilno, ad_soyad, dtar ve dyer sütunları ile verileri gösterir. Form alanları, Sicil No, Adı Soyadı, Doğum Tarihi ve Doğum Yeri için girişler sağlar. Ekle, Değiştir, Excelde Bak, Yenile ve Kapat butonları da mevcuttur.

sicno	ad_soyad	dtar	dyer
a2237	Fadime Çiçek	12.02.1962	Tokat
a4102	Ahmet Korkusuz	12.06.1965	İstanbul
a5206	Fatih Türkmen	01.07.1968	Manisa
b9622	Feride Limon	06.02.1969	Kahramanmaraş
c2196	Cenk Gürsel	15.11.1963	Erzurum
c4646	Mehmet Ali Çakır	02.02.1961	Ankara
d7889	Zeliha Sabuncu	05.06.1969	Kars

Ekle Değiştir Excelde Bak Yenile Kapat

Sicil No Adı Soyadı Doğum Tarihi Doğum Yeri

a5206 Fatih Türkmen 01.07.1968 Manisa

Tamam İptal

Program ilk çalıştığında formun alt kısmı görülmez. Bu özellik formun **Height** özelliği çalışma esnasında değiştirilerek sağlanmıştır. **Ekle** ve **Değiştir** butonlarına basıldığında alt taraf görünür. Bu kısım veri girişi yapmaya yardımcı olur. Değişiklik veya ekleme yapıldıktan sonra alt taraf tekrar görünmez hale getirilir.

7.4. ODBC VERİTABANLARINA ERİŞİM

Microsoft Jet veritabanı motoru ve DAO kullanarak ODBC verilerine erişim için iki yol vardır. Bunlar;

Microsoft Jet veritabanı motoru

ODBCDirect

Microsoft Jet önemli bir işlevselliğe sahiptir. Bağlantıların kurulması ve yönetilmesi, sorguların Veritabanı Sunucusu tarafından kabul edilebilir biçimlere dönüştürülmesi ve geri döndürülen verinin işlenmesi gibi önemli özellikler sağlar.

Bazı durumlarda Jet motorunu devreden çıkarıp DAO nesnelerinin doğrudan ODBC sürücüsüyle ilişki kurması gerekebilir. Bu ODBCDirect aracılığıyla gerçekleştirilir.

Jet veritabanı motorunun ve ODBCDirect yöntemlerinden herbirinin kendi çapında sağladığı avantajlar vardır. Aynı anda her ikisini birden kullanmak da mümkündür.

Microsoft Jet tarafından sağlanan ve ODBCDirect ile yapılamayacak olanlar:

- Değiştirilebilir birleştirme işlemleri. Birleştirme(Join) işlemi kullanılarak yaratılan Recordset nesneleri üzerinde değişiklik yapmak mümkündür.
- Bağlı tablo kullanabilme. Ağ üzerindeki bir tablo ile bağ kurmak bu tablonun genel bilgilerini bir yapı içinde tutmak demektir. Bu özellik sayesinde tabloya erişim daha hızlı sağlanabilir.
- FindFirst, FindNext, FindPrevious ve FindLast arama özelliklerini kullanabilme.
- Değiştirme sorguları için kısmi hata. Bu özellik hatanın olduğu yere kadar olan değişiklikleri kaydedebilme olanağı sağlar.
- Kullanıcı tarafından tanımlanabilen özellikler. Kullanıcı veritabanı nesnelerine yeni özellikler ekleyebilir.
- Veri özetleme için kullanılabilen SQL TRANSFORM sorgularını kullanabilme olanağı.
- Heterojen veri erişimi. Farklı veritabanlarından tabloları aynı anda kullanabilme.
- Program tabanlı DDL. ODBCDirect TableDef nesnesini sağlamadığı için tablo yaratmak mümkün değildir. Ancak SQL DDL kullanarak veritabanı yaratmak mümkündür.
- Bağlı kontrol nesneleri. ODBCDirect ile bağlı kontrol nesneleri kullanmak mümkün değildir. Jet motoru kullanarak bazı kontrol nesnelerini veritabanındaki alanlara bağlamak mümkündür.

Eğer bütün bu özelliklere ihtiyaç duyulmuyorsa ODBCDirect kullanılabilir. Ayrıca her iki yöntemin aynı anda kullanılabileceği de unutulmamalıdır.

ODBCDirect, ODBC API kullanarak DAO nesneleri aracılığıyla veritabanına doğrudan erişimi sağlar. ODBCDirect kullanarak kullanıcı bilgisayarında en az işlemler yapılarak veritabanı erişimi yapmak mümkündür. Sağladığı avantajlar:

- Doğrudan erişim. Sorgular doğrudan ODBC sunucusuna gönderilir.
- Düşük kaynak gereksinimi. Jet motoru yüklenmediği için kullanıcı bilgisayarındaki kaynak gereksinimi çok azdır.
- Sunucuya özel işlevselliği kullanabilme.
- Zamanuyumsuz sorgular. Bir sorgu bitmeden bir başkasını başlatabilmek mümkündür. Bu paralelliği artırır.
- Toplu halde değişiklik yapma olanağı. Değişikliklerin biriktirilip bir anda sunucuya gönderilmesi mümkündür.
- Daha esnek yordam çalıştırma.

7.4.1. ODBC VERİLERİNE MICROSOFT JET KULLANARAK ERİŞİM

ODBC veritabanlarına erişim için kullanılan bu iki yöntemin Visual Basic ile nasıl gerçekleştiğini görelim.

Sunucuya bağlanma.

ODBC sunucusuna bağlanmak için 3 yol vardır.

- Tablolara bağ oluşturma
- SQL sorgularını doğrudan sunucuya gönderme
- DAO kullanarak sunucuya doğrudan erişim.

Tablolara Bağ Oluşturma

Veritabanını açmaktan daha etkin bir yöntemdir. Böylece bu tablolar üzerinde Jet veritabanında yapıyormuş gibi sorgu yapmak mümkündür. Veri tabanını açmak ilk bakışta daha hızlı gibi görünebilir, fakat tablolara her erişim için veritabanına tekrar erişim gerekir. Bu yöntem tabloların genel bilgilerini bir yerde saklar ve veritabanına erişim miktarını azaltır.

Dikkat edilmesi gereken bir konu ise sunucu tarafından tablo yapısında yapılan değişikliklerin tutarsızlığa neden olabileceğidir. Çünkü bu yöntemle tablo yapısının bir kopyası kullanıcı bilgisayarında tutulur ve sorgularda tutulan bu yapı kullanılır. Değişiklik olduğunda tekrar bağ oluşturmak gerekir.

SQL sorgu sonuçlarına (view) bağ kurma ve yapay indeks oluşturma

Eğer sunucu SQL sorgu sonuçlarını tutan tablo yapılarını destekliorsa bu yapılara bağ kurmak mümkündür. Bu tablo yapıları için yapılan işlemler sunucuya aittir. Eğer sunucu bu yapılarda değişiklik yapmaya izin veriyorsa bu yapılar üzerinde indeks tanımlama da yapılabilir. Böylece sorgu sonuçlarına, herhangi bir satırı belirleyici özelliğe sahip bir alan kullanarak kolayca erişim sağlanır.

Örneğin PERSONEL veritabanı üzerinde “Yüksekokul mezunu personelin SicilNo, Ad ve Soyad, Görevi bilgilerinin listesi” sorgusunun sonucu **Yukse** adıyla bir tablo yapısında tutulsun. Bu yapıya kurulan bağ ise **Yukse1** olsun. Burada SicilNo hala tutanağı belirleyen bir alan olduğundan bunu bir indeks olarak kullanabiliriz. Bunun için aşağıdaki SQL deyiimi kullanılabilir.

CREATE UNIQUE INDEX Index1 ON Yuksek1(sicno)

Bu sunucuya gönderilen bir sorgu olmadığı için indeks sadece yapay bir indeks olarak kullanılır. Sunucu üzerinde böyle bir indeks yoktur.

DAO kullanarak ODBC verilerine erişim

DAO kullanarak ODBC verilerine erişmek için;

- 1-** Windows Control Panel penceresinden 32bit ODBC seçeneğini kullanarak ODBC veri kaynağı ile ilgili ayarları yapın ve bir veri kaynağı ismi (DSN) atayın
- 2-** Veri kaynağını temsil eden bir bağlantı (connect) metnı tanımlayın.
- 3-** Yukarıda tanımlanan bağlantı metnini kullanarak TableDef nesnesi yaratın.
- 4-** Bu TableDef nesnesini TableDefs kümesine ekleyin.
- 5-** Bu bağlı tablo üzerinde bir Recordset açın. Bu Recordset yapısı ODBC veri kaynağının verilerini barındırır. Recordset nesnesinin özellikleri kullanılarak bu veriler üzerinde işlem yapmak mümkündür.

Şimdi bu adımları açıklayalım.

ODBC Data Source Menager kullanımı:

- Control panel penceresinden 32bit-ODBC seçeneğini seçin.
- Add düğmesine basıp bir ODBC sürücüsü seçin.(Ms SQL server)
- Bir DSN adı yazın(Personel)
- Veritabanı ile ilgili bir açıklama yazın.(abc kurumunun personeli)
- Sunucunun adını yazın.(vtSunucu)
- Veritabanının ağ adresini yazın.

Bağlantı metnini hazırlama

Bu metin birbirinden noktalı virgülle ayrılmış ODBC sürücüsünün bağlantıyı sağlaması için kullanılan birtakım değerlerden oluşan bir satırdır. İlk değer "ODBC;" şeklinde olmalıdır. Örneğin;

ODBC;UID=mustafa;pwd=mmtt12;DSN=Personel

Bu satırda bazı bilgiler eksik kalırsa ODBC sürücüsü eksik bilgilerin girişi için bir dialog kutusu çıkarır.

3,4,5. Bağlı tablo oluşturma.

Aşağıdaki fonksiyon bağlı tablo oluşturarak personel veritabanında "Ankara" doğumlu personel sayısını döndürür.

```
Public Function ankSay() As Long  
  
Dim db As Database, rs As Recordset  
  
Dim strBag As String, tdf As TableDef  
  
Dim say As Long  
  
strBag = "ODBC;DSN=Personel;UID=mustafa;" & _  
"PWD=mmtt12;DATABASE=Personel"  
  
Set db = OpenDatabase(App.Path & "\Personel.mdb")  
  
On Error Resume Next  
  
db.TableDefs.Delete "bPers"  
  
db.TableDefs.Refresh  
  
On Error GoTo phata  
  
Set tdf = db.CreateTableDef("bPers")  
  
tdf.Connect = strBag  
  
tdf.SourceTableName = "personel_bil"
```

```
db.TableDefs.Append tdf  
  
db.TableDefs.Refresh  
  
Set rs = db.OpenRecordset("bPers", dbOpenForwardOnly)  
  
Do Until rs.EOF  
  
If Trim$(rs!dyer) = "Ankara" Then say = say + 1  
  
rs.MoveNext  
  
Loop  
  
rs.Close  
  
db.Close  
  
ankSay = say  
  
Exit Function  
  
phata:  
  
MsgBox "Error " & Err & ": " & Error  
  
Exit Function  
  
End Function
```

Bu örnekte PERSONEL_BIL tablosu forward-only bir Recordset yapısıyla açılmıştır. Bağlı ODBC tablolarının kullanım açısından daha önce kullanılmış olan Jet veritabanı motoru tablolarından bir farkı yoktur. Örneğin aşağıdaki sorgu bağlı bir tablo üzerinde çalıştırılabilir.

```
Dim db As Database, qdf As QueryDef  
  
Set db = db1 'Kullanılmakta olan bir veritabanı  
  
Set qdf = db.CreateQueryDef("bPers")
```

```
qdf.SQL = "DELETE FROM bPers WHERE dycer = ' Ankara'"  
qdf.Execute
```

Bu kod daha önce yaratmış olduğumuz bağlı tablodan “Ankara” doğumlu olan personeli siler.

SQL sorgularını doğrudan sunucuya gönderme

ODBC veritabanı üzerinde SQL sorguları Jet veritabanı motoru kullanılarak da gerçekleştirilebilir fakat doğrudan sunucu üzerinde çalıştırılmaları işlemlerini daha hızlı yapılmasını sağlar.

Bu sorgular herhangi bir yorumlama yapılmadan doğrudan sunucuya gönderilir. Sorgular iki kesimden oluşur, SQL sorgu deyimi ve ODBC bağlantı metni. SQL deyimleri sunucunun kabul ettiği türden olmalıdır, Visual Basic deyimleri içermemelidir.

Avantaj ve dezavantajları:

- Sunucunun işlem miktarı artar, ağ trafiği azalır.
- Sunucu tarafından sağlanan özel işlevleri kullanabilirler.
- Sunucudan gelen mesajların listesini tutarlar.
- Sunucu üzerindeki farklı veritabanlarını birleştirmenin tek yolu bu tür sorgular kullanmaktır.
- Çok sayıda silme ve değiştirme yapan bu tür sorgular normal sorgulardan çok daha hızlıdır.
- Geri döndürülen Recordset yapısı ancak Snapshot olabilir.
- Değişiklik yapan bir sorgunun başarısızlığı kısmi olamaz.

Bu tür sorgular aşağıdaki gibi bir kodla çalıştırılabilir.

```
Dim db As Database, qdf As QueryDef  
strBag = "ODBC;DSN=Personel;UID=mustafa;" & _  
"PWD=mmtt12;DATABASE=Personel"  
Set db = db1 'Kullanılmakta olan bir veritabanı  
Set qdf = db.CreateQueryDef("")  
qdf.SQL = "DELETE FROM bPers WHERE dycer = ' Ankara'"
```

```
qdf.Connect = strBag  
qdf.Execute
```

Burada verilen Connect özelliği sorgunun doğrudan ODBC sunucusuna gönderilmesini sağlar.

Doğrudan sunucuya gönderilen sorgular aşağıdaki şekilde de çalıştırılabilir.

```
strBag = "ODBC;DSN=Personel;UID=mustafa;" & _  
"PWD=mmtt12;DATABASE=Personel"  
Set db = OpenDatabase(App.Path & "\Personel.mdb")  
db.Execute "DELETE FROM bPers WHERE dyer = ' Ankara'"  
db.Close
```

7.4.2. ODBC VERİLERİNE ODBC DIRECT KULLANARAK ERİŞİM

ODBCDirect kullanmak için önce bir ODBC Workspace yaratmak gerekir. Bu CreateWorkSpace metodu kullanılarak yapılabilir. Bu işlem aşağıdaki gibi yapılabilir.

```
Dim wks As Workspace  
Set wks = DBEngine.CreateWorkspace("ODBCWks", "mustafa", "mmtt12", _  
dbUseODBC)
```

ODBC Workspace yaratıldıktan sonra yapılması gereken ODBC veri kaynağıyla bağlantı kurmaktır. Bağlantı OpenDatabase metodu ile yapılır. Parametre olarak uygun bir bağlantı metni verilir. Veri kaynağına bağlanmak için iki yol vardır. Bunlar Database nesnesi yaratmak ve Connection nesnesi yaratmaktır. Connection nesnesi ODBC verilerine erişimde sağladığı bazı olanaklar sayesinde verimlilik sağlar. Bu olanaklar;

- **Zamanuyumsuz bağlantı.** Bir uygulama bağlantının kurulmasını beklemek zorunda değildir.
- **Zamanuyumsuz sorgu işleme.** Bir sorgunun bitmesini beklemeden başka bir sorgu işletebilme olanağı.
- **QueryDef nesnesi.** ODBC veri kaynağı üzerinde yapılan sorguları bir yapı içinde tutmayı sağlar. Böyle bir olanak Database nesnesi kullandığında yoktur.

Bu olanaklardan yararlanmak için kullanılan Connection nesnesi aşağıdaki gibi kullanılır.

Set **variable = workspace**.OpenConnection (**dsName, prompt, readonly, connect**)

Bu metodun kullanışı şu şekilde örneklenebilir.

```
Dim cnn As Connection, strCn as String  
  
strCn = "ODBC;dsn=Pubs;database=Pubs;uid=sa;pwd=;"  
  
Set cnn = OpenConnection("", dbDriverNoPrompt, False, strCn)
```

Connection nesnesi yaratıldıktan sonra bu nesne üzerinde Recordset yapısı tanımlamak ve üzerinde sorgu yapmak mümkündür.

ODBCDirect yöntemiyle aynı ODBC veri kaynağı üzerinde hem Connection hem de Database nesnesi tanımlamak mümkündür. Database nesnesi ancak Microsoft Jet veritabanı motorundan yararlanılacaksa kullanılmalıdır.

7.5. VERİTABANI PERFORMANSINI ENİYİLEŞTİRME

Yerel ve uzak veritabanı sistemleri arasında önemli farklar olduğundan dolayı performansları da farklıdır. Yanlış tasarlanmış bir Client/Server veritabanı ihtiyaçlara cevap vermez. Bu sebepten dolayı client/server yapısında veritabanı tasarlanırken çok dikkatli olunmalıdır.

Sorguların hızlandırılması

Uzak bir sistemdeki sorguların performansını artırmak için yapılabilecek en iyi şey sorguların olabildiğince fazla kısmını sunucu üzerinde çalıştırmayı sağlamaktır.

Microsoft Jet veritabanı motoru verilen bir sorgu için ön işlemleri yaparak bunu ODBC veri kaynağının desteklediği türden sorgular haline getirir. Eğer bir sorgu bir çok deyim içeriyorsa bir ön işlemeye ihtiyaç vardır. Bu ön işleme yerel olarak yapılır. Sorguları iyileştirmek için sorgularda genel olarak sağlanan işlevler kullanılmalıdır.

Genel olarak sağlanmayan işlevler şunlardır.

- Basit bir SQL deyimi ile açıklanamayan işlemler.
- Microsoft Jet motorunun sorgular için getirdiği ekler.
- Microsoft ürünlerine has fonksiyon ve işlemler.
- Kullanıcı tarafından tanımlanan fonksiyonlar.
- Farklı veri türlerini bir arada kullanan deyimler.
- Yerel ve uzak tablolar arasında birleştirme işlemleri.

Eğer işlem sunucu tarafından destekleniyorsa, doğrudan sunucuya gönderilir, desteklenmiyorsa yerel olarak işlenir.

Daha az veri istemek

Ne kadar çok veri istenirse ağ trafiği o kadar artacağından erişim hızı gittikçe düşer. Artan ağ trafiği sunucunun işlem yükünü de artırır. Bu istenmeyen durumlara engel olmak için;

- Olabildiğince az tutanak istenmelidir.
- Uzak tutanaklardaki alanlardan olabildiğince azı seçilmelidir.
- Olabildiğince az bağlı kontrol nesnesi kullanılmalıdır.

Alanları sadece istendiğinde gösterme

Form içinde veritabanı tablosunun alanları gösterilirken bazen sadece bir alanın tüm değerleri kullanılırken, bazen de bir tutanağın tüm alanlarına ihtiyaç duyulur. Bu tür durumları ele almak için şu yöntemler kullanılabilir.

- Form üzerinde sık kullanılmayan alanlara bağlı nesnelerin **Visible** özelliklerini gerekmediği zaman False yaparak kullanılmaz hale getirmek
- En önemli alanları bir forma yerleştirip daha az önemsiz alanları başka bir forma eklemek ve bu formu göstermek için asıl forma bir buton yerleştirmek.

Verileri birden fazla form için kullanılmak üzere istemek

Genellikle uygulamalar, birden fazla form kullanır ve bu formlar uzak sistemlerdeki aynı verilere erişir. Bu tür durumlarda performans iyileştirmek için şunlar yapılabilir.

- Eğer bir tablo hiç değişmeyen birtakım bilgiler içeriyorsa bunlar yerel bir veritabanı üzerine kaydedilmelidir.
- Sık değişmeyen veriler için de yine yerel bir kopya tutulmalı, verilerde değişiklik olduğunda bu veriler güncellenmelidir.
- Sık değişen verilerin de bir kopyası yerel veritabanı üzerinde tutulmalı ancak verilerin güncellenmesi için otomatikleştirilmiş bir mekanizma kullanılmalıdır.

İşlevselliği daha az kullanma

Veritabanının performansının iyileştirilmesi için, Microsoft Jet veritabanı motoru tarafından sağlanan bazı işlevlerin kullanılmasından sakınmak gerekir.

- Eğer veriler değiştirilmeyecekse, Dynaset türü Recordset yerine Snapshot türü Recordset tercih edilmelidir. Doğrudan sunucuya gönderilen sorgular için forward-only türü Recordset yapılarını kullanmak hızı artırır. Tutanak boyları uzun olduğunda dynaset türünü kullanmak daha elverişlidir. Çünkü bu durumda dynaset yapısı sadece birincil anahtarları getirirken, diğerleri tüm tutanağı getirir.
- Birden fazla tablodan verileri içeren bilgileri form üzerinde göstermenin iki yolu vardır. Sonuçların hepsi tek bir form üzerinde gösterilir, veya başka formlar kullanılır. Birden çok tabloyu içeren bir sorgu için tek bir form kullanılırsa sorgu bir defada sunucuya gönderilir. Alt formlar kullanılırsa her bir alt form için başka bir sorgu gönderilir. Alt formlar veri günleme yapılırken avantaj sağlar. Veri günlemenin yapılmadığı durumlarda alt form kullanılmamalıdır.

Uzak verilerin geçici Recordset yapılarında tutulması

Günleme ve silme işlemleri için performansını artırmanın en basit yolu, uzaktaki tabloya tablonun ne zaman değiştiğini belirten bir alan yerleştirmektir. Bu alan sunucu tarafından yönetilir ve tutanak günlendiği anda günlendir. Bu alanı doğrudan okumak mümkün olmasa bile Microsoft Jet bu alan sayesinde tutanağın değişip değişmediğini algılayabilir. Böyle bir alanın olmadığı durumlarda bir tutanağın değişip değişmediğini anlamak için tüm tutanağın karşılaştırılması gerekir. Bir tabloya böyle bir alan aşağıdaki SQL deyiimiyle eklenebilir.

ALTER TABLE UzakTablo ADD SonDegZ TIMESTAMP

Bu deyimle UzakTablo adlı tabloya SonDegZ adlı, tablonun son değiştiği zamanı belirten bir alan eklenir.

Uygulama performansını artırmanın bir diğer yolu da uzaktaki veriyi geçici bellek alanlarında tutmaktır.

Bir Recordset tanımlayıp bunu yerel bilgisayara kopyalamak için Recordset yapısının CacheStart, CacheSize özellikleri ve FillCache metodu kullanılabilir. Verilerin bir kopyasını yerel bilgisayara almak bu Recordset yapısı üzerinde yapılacak sorguların hızını artırır.