

# VERİ TABANI YÖNETİM SİSTEMLERİ II



YRD. DOC. DR. ZEHRA ALAKOÇ BURMA

## İÇİNDEKİLER

VERİ TABANI YÖNETİM SİSTEMLERİ-II.....	
1. GİRİŞ.....	<u>3</u>
1.1. Veri Tabanı Yönetim Sistemleri Yapısı.....	<u>4</u>
1.2. Veri Tabanı Yönetim Sistemleri İstemci/Sunucu Mimari Yapısı.....	<u>6</u>
1.3. RESULTSET Kavramı.....	<u>9</u>
2. SQL PROGRAMLAMA, VERİ TİPLERİ ve DEĞİŞKENLER.....	<u>10</u>
2.1. SQL Programlamanın Avantajları.....	<u>10</u>
2.2. Veri Tipleri (Data Type) ve Değişkenler.....	<u>11</u>
2.2.1. Oracle Veri Tabanı Veri Tipleri.....	<u>11</u>
2.3. Veri Tabanı Programlarında ve SQL de Değişken Tanımlama.....	<u>12</u>
2.4. Değişkenlere Değer Atamak.....	<u>13</u>
3. SQL PROGRAMLAMA BLOKLARI ve AKIŞ DENETİMİ.....	<u>15</u>
3.1. SQL Programlama Bloklarının Yapısı.....	<u>15</u>
3.2. SQL Programlamada Akış Kontrolleri.....	<u>16</u>
3.2.1. PL/SQL Akış Kontrolleri.....	<u>16</u>
3.2.2. T-SQL Akış Kontrolleri.....	<u>19</u>
3.3. SQL Geçici Tabloları.....	<u>22</u>
4. SQL PROGRAMLAMADA PROCEDURE ve FUNCTION.....	<u>23</u>
4.1. Procedure (Prosedür) ve Özellikleri?.....	<u>23</u>
4.2. Procedure (Prosedür) Oluşturma ve Değer Döndürme.....	<u>24</u>
4.2.1. PL/SQL Procedürleri ve Fonksiyonları.....	<u>24</u>
4.2.2. T-SQL Procedürleri.....	<u>28</u>
5. SQL PROGRAMLAMADA CURSOR (İMLEÇ) ve TRIGGERS (TETİKLEMELER).....	<u>34</u>
5.1. Cursor (imleç) ve Özellikleri.....	<u>34</u>
5.1.1. PL/SQL Cursors (İmleçleri).....	<u>34</u>
5.1.2. T-SQL Cursors (İmleçleri).....	<u>36</u>
5.2. Triggers (Tetiklemeler).....	<u>38</u>
6. SQL PROGRAMLAMADA TRANSACTION (İŞLEMLER) ve HATA DURUMLARI.....	<u>40</u>
6.1. Transaction (İşlemler) ve Özellikleri.....	<u>40</u>
6.2. Hata Durumları.....	<u>42</u>
7. FORMLAR ve ACCESS PROGRAMINDA FORM OLUŞTURMA.....	<u>44</u>
7.1. ACCESS ile FORM Oluşturma.....	<u>45</u>
8. RAPORLAR VE ACCESS PROGRAMINDA RAPOR OLUŞTURMA.....	<u>55</u>
8.1. ACCESS ile RAPOR Oluşturma.....	<u>55</u>
9. FORMLAR ve ORACLE FORMS PROGRAMINDA FORM OLUŞTURMA.....	<u>61</u>
9.1. ORACLE FORMS ile FORM Oluşturma.....	<u>61</u>

10.	FORMLAR ve VISUAL BASIC PROGRAMINDA FORM OLUŞTURMA .....	<u>67</u>
10.1.	VeriTabanı Tasarımı ve Visual Data Manager Kullanımı .....	<u>67</u>
10.2.	Visual Database Bileşenleri .....	<u>71</u>
10.3.	Visual Basic Ortamına Otomatik Veri Formu Aktarmak .....	<u>74</u>
10.4.	Data Form Wizard ile Form Oluşturulması .....	<u>76</u>
10.4.1.	Tek Tablo İle Form Oluşturulması .....	<u>77</u>
10.4.2.	Master / Detail ile Form Oluşturulması .....	<u>82</u>
11.	RAPORLAR ve VISUAL BASIC PROGRAMINDA RAPOR OLUŞTURMA....	<u>87</u>
11.1.	Data Report Designer ve Özellikleri .....	<u>87</u>
11.2.	Data Report Designer Kontrolleri .....	<u>88</u>
11.3.	Data Report Designer 'da Bir Raporun Yapısı .....	<u>88</u>
11.4.	Data Report Designer ile Rapor Oluşturulması .....	<u>90</u>
B.	KAYNAKLAR .....	<u>96</u>

## VERİ TABANI YÖNETİM SİSTEMLERİ-II

### 1. GİRİŞ

Veri tabanı, bir kuruluşun uygulama programlarının kullandığı operasyonel verilerin bütünüdür. Veritabanı Yönetim Sistemleri, verilerin fiziksel hafızadaki durumlarını, kullanıcıların erişimlerini düzenleyen sistemlerdir. Günümüzde, bir çok alandaki veri işlemlerinde pek çok Veri Tabanı Yönetim Sistemleri programları yaygın olarak kullanılmaktadır. Oracle, Sybase, SQL Server, Informix gibi birbirinden farklı isimler adı altında anılan bu programlar için bir çok nesne birbiri ile aynı temel işlevi yerine getirmekte olup, yaklaşık olarak aynı teorilere dayanarak çalışırlar.

Açılımı “Structured Query Language” yani “Yapısal Sorgulama Dili” olan SQL, veritabanı işlemleri ile ilgili komutlardan oluşan bir dildir. Bu dil tüm veritabanı programlarında kullanılabilir. Bu dil ile veri tabanı üzerinde; veritabanının kendisini oluşturmak, tablo, indeks, kullanıcı oluşturmak gibi komutlar ve kayıt ekleme, silme, düzeltme gibi işlemler yapılabilir. SQL bir dildir; ancak bir programlama dili değildir. Program geliştirme aşamasında SQL 'den faydalanılır, ancak tek başına SQL bu iş için yeterli değildir. PHP, Asp, Visual Basic, Delphi, C, C++ gibi bir çok programlama dili SQL komutlarını desteklemektedirler. Yani bir program geliştirme aşamasında; SQL komutlarını bilmek gerekmektedir ama SQL tek başına bir programlama dili olmadığı için ayrıca bir programlama diline de ihtiyaç duyulmaktadır.

SQL dilindeki komutlar Pascal, C, Visual Basic, Delphi ve benzeri dillerdeki fonksiyon ve prosedür oluşturarak bir program yazmaktan biraz farklıdır. Yani kullanıcı SQL kullanırken fonksiyon ve prosedür yazamaz. Yine SQL kullanımında şartlı ifadeler ve dallanmalar bulunmaz. Yani kullanıcı diğer programlama dillerindeki If, Case, next, do gibi ifadeler kullanamaz. Şartlı ifadeler, döngüler, karşılaştırmalar SQL 'de bulunmaz.

SQL'de kullanılamayan procedür, fonksiyon, şartlı ifadeler, döngüler ve karşılaştırmaların eksikliğini giderebilmek için; Oracle PL/SQL (Programming Language/SQL), MS SQLServer ve Sybase T-SQL (Transact SQL) dilini geliştirmiştir. Oracle tarafından kullanılan PL/SQL komutları ile T-SQL bir çok noktada hemen hemen aynıdır ve SQL'de kullanılamayan procedür, fonksiyon, şartlı ifadeler, döngüler ve karşılaştırmalar kullanılabilir. PL/SQL ve T-SQL if,case,for..next gibi programlama için gereken işlemleri kullanmayı olanaklı kılar.

Görüldüğü gibi SQL karar yapıları, döngüler ve benzeri gibi bir programlama diline özgü yetilerden yoksundur. Ancak bir çok VTYS'de bu yetiler Transact-SQL veya PL/SQL gibi dil tanımları ile bir noktaya kadar desteklenmiştir. Verilerin hacmi arttıkça Oracle, Sybase, Informix gibi daha gelişmiş VTYS'lere ihtiyaç duyulur.

PL/SQL ve T-SQL sadece içerisinde SQL komutları kullanılabilen bir dildir. Yani SQL'in yapısını değiştirmemiştir. Komut modunda yazılan bir SQL cümlesi alınıp PL/SQL ve T-SQL blokları arasına yazılabilir. Veri tabanı programları tüm uygulamalarda SQL kullanmayı esas almıştır. Kullanıcı veri tabanı ürünlerini kullanarak yaptığı tüm işlemlerin arkasında SQL komutlarını çalıştırır.

## 1.1 VERİ TABANI YÖNETİM SİSTEMLERİ YAPISI

Günümüzde en yaygın olarak kullanılan veritabanı; İlişkisel Veritabanı (Relational Database) yaklaşımında olan veritabanıdır. Bu veritabanı yaklaşımı, verileri normalizasyon kuralları çerçevesinde tablolara ayırt etmeyi ve bu tablolar arasında bir birincil anahtar ve bir yabancı anahtar üstünden ilişki kurmayı öngörür. Oracle, MS SQL Server, Sybase, Informix, MySQL, Postrage, Access, Tamino, BerkeleyDB VTYS programlarının bir çoğu bu türden Veritabanı Yönetim Sistemleri (VTYS) 'dir.

Yoğun olarak kullanılan veritabanları; veri üretilen yerlerde iki genel amaca yönelik olarak kullanılırlar. Bunlardan birincisine, optimize edilmiş sistemler OLTP (Online Transaction Processing) adı verilir ve üretilen verilerin anlık olarak saklanması işlemini yapar. Bu türden işlemler için OLTP sistemlerde; verilerde sürekli olarak değişiklikler, eklenmeler ve silinmeler olabilir. İkincisine; OLAP (Online Analytical Processing) adı verilir ve daha çok raporlama ve karar destek amacı ile kullanılır. Bu türden sistemler aracılığıyla veri ambarı ve datamark gibi yapılar kullanılarak yoğun bir şekilde üretilmiş verilerin analizleri ve raporları oluşturulur. Böylece tüketici ve satış eğilimleri, üretim maliyetleri gibi konularda kullanılacak sonuçlar elde edilir.

SQL Server, Oracle ve Sybase gibi bir çok VTYS hem OLAP hem de OLTP sistem olarak ayarlanabilirler. Ancak; bir veritabanı yöneticisinin ilgili ayarlamaları yapması gerekir. Veritabanının kurulumu, yedeklenmesi, replication gibi düzenli bakım gerektiren işlemlerinin gerçekleştirilmesi de ayarlamalar ile birlikte, kullanıcı yönetimi ve yetkilendirme gibi işler '**Veritabanı Yönetimi**' olarak adlandırılmaktadır. Bu işleri yapan kişi ise '**Veritabanı Yöneticisi**' olarak adlandırılır.

Bir veritabanı programcısı ise, tabloların neler olacağına, hangi constraintler ile birbirlerine bağlanacağı, artan kayıt sayılarına bağlı olarak uygulamanın hız problemini aşmak için hangi indekslerin tanımlanacağı; view, stored procedure gibi yapılarla veritabanı uygulaması ve veritabanı arasında güvenli bir erişim sağlama, Constraint'lerin yetersiz kaldığı durumlarda trigger gibi yapılarla veri bütünlüğünü sağlama gibi konulara hakim olan kişidir. Veritabanı programcısının bir uygulamayı ortaya koyabilmesi için ayrıca bir API ile veritabanına erişimi ve bu API 'nin kullanılabileceği ortamı programlamayı da öğrenmesi gerekir. Bütün bunlar ciddi birikimler gerektirir.

Veritabanı programları genellikle istemci-sunucu (Client/Server) temel mimarisi üstünde çalışırlar. İstemci/sunucu sistemler iş hayatında, üniversitelerde, resmi kurumlarda, bilim, bankacılık, mühendislik, tıp ve boş zamanları da içine alan tüm alanlarda kullanılmaktadır.

Veritabanları Transaction Log adı verilen dosyalar içerir. Bu dosyalar aktif olarak RAM' de çalışır. Veritabanında bir değişiklik yapıldığında; bilgiler önce Transaction Log dosyasına yazılır daha sonra veritabanı tarafından hata kontrolü yapılarak diske aktarılır. Transaction Log dosyasının boyutu veritabanı boyutunun %10 - %25 aralığında olmalıdır.

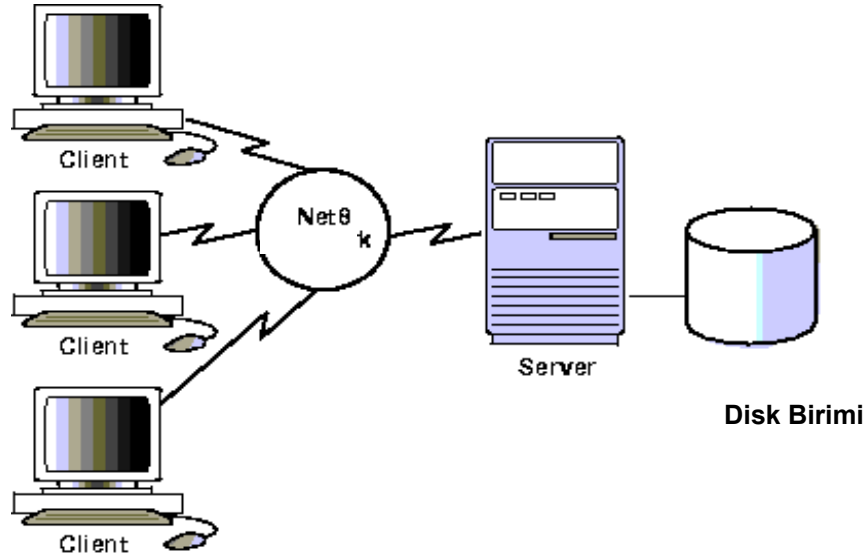
Veritabanı dizaynında dikkat edilecek en önemli noktalardan biriside veritabanı optimizasyonudur. Veritabanlarının performansının kullanıcı tarafından değiştirilmesine olanak sağlayan Windows altında çalışan kullanıcıyı kolay görsel özel programları vardır. Bunlarla optimizasyon sağlanabilir. Bu programlar yoksa, bu iş performansla ilgili SQL komutları ile de yapılabilir. SQL'in iyi öğrenilmesi şarttır.

Optimizasyon için dört teknik kullanılır:

- 1. Veri dosyalarının farklı disklerde oluşturulması :** Veri dosyalarının farklı disklerde oluşturulması ile performans artışı sağlanabilir. Dosya, tablo ve indexlerin farklı disklerde tutulması ile aynı zaman diliminde farklı indexlerden okuma yapılabilir.
- 2. Veri grupları oluşturmak :** Bir veri tabanı yaratıldığında otomatik olarak dosya grubu (file group) oluşturulur. Grup oluşturulması geniş veritabanlarında yedekleme işlemlerinin daha kısa zamanda ve daha az disk kapasitesi kullanılarak yapılmasını sağlar. Bu dosya grubu veritabanının Primary veri dosyasını da içerir. İkinci veri dosyası da (Secondary Data Files) bu gruba eklenebileceği gibi yeni kullanıcı açılarak da ikinci veri dosyası oluşturulabilir. İkinci veri dosyasının, bölümlendirilmiş bir diskte oluşturulması performans artışı sağlar. Veri grupları sadece veri dosyalarının saklanması için kullanılabilir. Transaction Log dosyaları grup içerisinde saklanmaz ve Transaction Log çoğullanarak (multiple), diskin diğer bölümlerinde saklanabilir. Böylece performans artışı sağlanabilir.
- 3. RAID sisteminin oluşturulması :** RAID sistemi ile birden fazla diskin sistemde kullanılması disk I/O performansını artırır. Veritabanlarının bir çoğu RAID 0, 1, 5 ve 10 seviyelerini desteklemektedir. RAID 0 seviyesinde, asıl ve yedek olmak üzere birbirinin aynı iki disk vardır ve bilgiler asıl ve yedek diske de aynı anda ve kontrol yapılmadan yazılır. RAID 1 seviyesinde, 1. Diske yazma yapılır ve 1.diskten ikinciye bilgiler aktarılır (mirror). RAID 5 seviyesinde, parity biti kullanılarak bilgi doğruluğu kontrol edilir. Bir diskte hata oluştuğunda, diğer parity biti kullanılarak server bilgiyi yeniden düzenler. Veri okunmasında server performansı artarken yazma işleminde hız düşer. Çünkü server parity bilgilerinin de yazımı için zaman ayırır. RAID 10 ise RAID 1 ve RAID 0' ın bir kombinasyonu olarak düşünülebilir. RAID 10' da 8 disk kullanılır. İlk dört disk yedekleme işlemi için kullanılır. Diğer dört diske ise birinci gruptan mirror yapılır.
- 4. Bölümlendirilmiş disklerin Transaction Log ve Tempdb için kullanılması :** Veritabanı Transaction Log dosyalarını her yazma işleminden önce kullanır. Diskten okuma işlemlerinde disk kafasının belirtilen iz ve sektöre konumlanması zaman alıcı faktördür. Bunun en aza indirilmesi ile zaman kaybı azalır.Transaction Log' a ayrı bir disk kullanılması ile disk okuma- yazma kafasının yazma konumu belirlenir. Tempdb veritabanı tabloların sıralanması, indexlerin düzenlenmesi gibi işlemlerde kullanır. Tempdb veritabanının, kullanıcı veritabanlarından ayrılması da tavsiye edilir. Tempdb' de ki bilgiler kalıcı değildir. Her açılışta eski bilgiler temizlenir.

## 1.2 VERİ TABANI YÖNETİM SİSTEMLERİ İSTEMCİ/SUNUCU MİMARİ

### YAPISI



**Şekil 1.2.1.** İstemci / Sunucu Yapısı

Yukarıdaki şekilde; bir server (ana bilgisayar) ve networklerle birbirine bağlı client bilgisayarlar bulunmaktadır. Client bilgisayarlar aptal terminal veya PC olabilirler. Disk biriminde ise Veritabanı programı, database, tablolar, index, view, procedure, trigger gibi veritabanı elemanları ve veritabanına yazılan bilgiler bulunur. İstemci bazı hizmetler için istekte bulunurken, sunucuda bu hizmetleri üreterek yanıt verir.

Öncelikle, gerçek veri depolama ve veritabanı sorgulama sunucular tarafından gerçekleştirilmektedir. İstemciler, istekte bulunmakta, yanıtları işlemekte ve gösterime getirilmektedir. Burada client 'lar Server'daki veritabanına erişmeye ve yeni bilgi kaydetmeye, veya bilgiler üzerinde değişiklik, silme, listeleme gibi işlemler yapma hizmeti isteğinde bulunurlar ve server 'da bu isteklerine cevap verir.

İstemci/Sunucu yapısının avantajları arasında, merkezileşmiş yönetim, güvenlik, veri bütünlüğü, paylaşılan veri, yazıcı gibi paylaşılan kaynaklar, daha az maliyet, en düşük yineleme ve en düşük uyumsuzluk vardır.

VTYS ler istemci/sunucu modeline şu iki nedenle çok iyi uyumluluk gösterirler:

- Büyük veritabanlarının, çoklu kullanıcılar tarafından erişilmesi gereksinimi vardır. Bir büyük veritabanı, tipik olarak bir çok kullanıcı tarafından paylaşılmaktadır. Büyük bir veritabanının, kişisel kopyalarına her kullanıcının sahip olmasının hiç bir anlamı yoktur. Böyle olursa; herkes kendi kopyası üzerinde işlem yapacağı ve girilen her bilgi kendi kopyası üzerinde kalacağı için verilerin bütünlüğü bozulur. Bu yüzden büyük bir veritabanının bir kopyasının merkezi denetimde tutulması büyük anlam taşır. Bütün bunlardan sonra, istemcilerdeki kullanıcılar merkezi bir VTYS 'i (Veri Tabanı Yönetim Sistemini) paylaşabilirler.

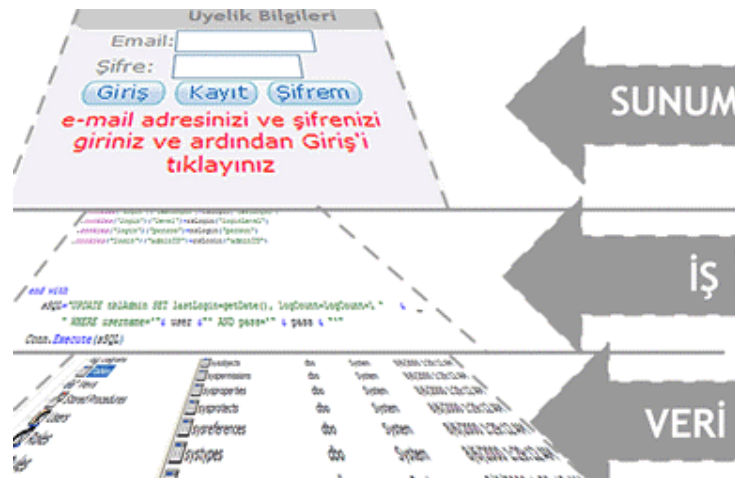


- Veritabanlarından geri döndürülen bilgilerin, diğer veritabanındaki bilgilerle ve diğer uygulamalar ile bütünleştirilmesi gereksinimi vardır. Personel veritabanı bilgisinden elde edilen personeller, muhasebe veritabanında maaş ve ödemeler için kullanılabilirler. Bir sorguda elde edilen veriler bir hesap tablosuna yerleştirilebilir veya diğer verilerle bütünleştirilebilir. Örneğin, sorgu sonucu elde edilen parça fiyatları, bir ürün maliyetleri planıyla bütünleştirilebilir. Bir departmanın bütçesi, bir hesap tablosunda oluşturulup bir veritabanına girilebilir ve orada tüm diğer departmanların bütçeleriyle birleştirilebilir. Bir kullanıcının geliştirdiği PC uygulaması bütün borçlu hesapların yer aldığı bir veritabanını sorgulayabilir. Buradan döndürülen bilgilerde, ödeme emri oluşturan mektuplarla birleştirilebilirler. Ayrıca istemciler veritabanı bilgilerini; Word, Excel, Power Point gibi Office programlarında yani masaüstü uygulamalarda birleştirme gereksinimi duyabilirler.

Client/Server mimarisindeki bir veritabanı yazılımı terminallere sadece yetki verdiği işlemleri yaptırır ve ancak bu işlem sağlıklı sonuçlanınca kendi üzerindeki data dosyasına yazar. Dolayısıyla kullanıcılar terminallerden data dosyasını bozacak faaliyetlerde bulunamazlar. Client/Server yapıdaki bir veritabanı sistemi elektrik kesintilerinde işlem yapılan veri kaydının, en azından o anda yapılan değişiklik ve işlemin sonucu dışındaki en son halini korur.

Veritabanı programlarının genellikle istemci-sunucu (Client/Server) temel mimarisi üstünde çalıştığı daha önce belirtilmişti. Ancak bu temel mimari kullanılırken, uygulama üç ayrı katmana ayrılarak incelenir.

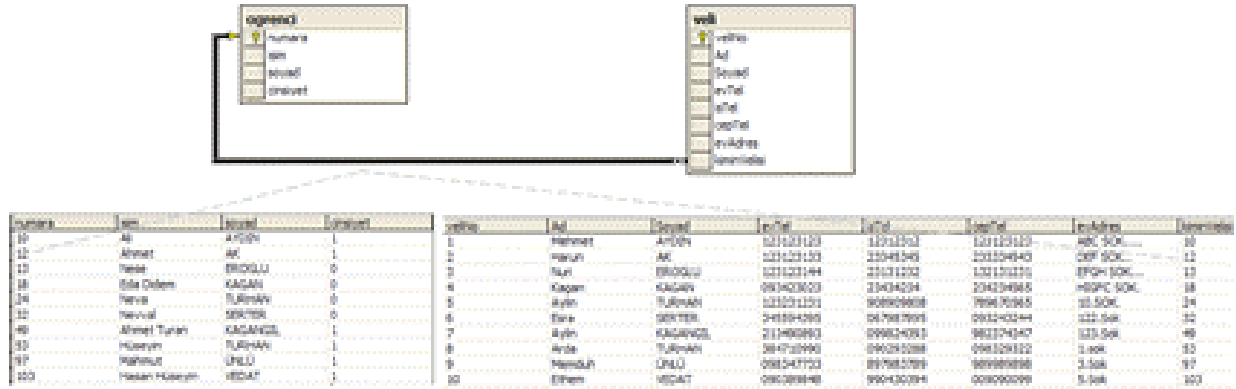
Veritabanı programlamanın konsepti 3 katman mimarisidir. Bunlar sunum, iş ve veri katmanlarından oluşur. Bütün uygulama sunucu tarafında ise N-Tier, adını alır. İş katmanı sunucuda ise akıllı sunucu; istemcide ise **akıllı istemci** (Intelligent Client) uygulama ismini alır. Geçici bir süre için veri katmanını da istemcide tutup, bir bağlantı sağlandığı anda sunucudaki veri katmanı ile senkronize olabilen istemciye de **Smart Client istemci** denir.



Şekil 1.2.2. 3 Katman Mimarisi



Örneğin; bir okulun öğrencilerinin aranabildiği bir Web Sitesi yapmak isteyelim. Bu sitenin büyük bir veritabanı kullanılarak, ASP'de programlanmasını planlayalım. Bu uygulamayı gerçekleştirmek için gerekli şeyleri gözden geçirerek konsepti anlamaya çalışalım.



**Şekil 1.2.2. Projede Tasarlanan Tablolar**

Verilerin tutulduğu veritabanı için, veri katmanı deyimi kullanılır. Çünkü bütün veriler, sorgular, tablolar vb. her şey burada saklanır. Uygulamanın en alt katmanını oluşturur. Projenin büyüklüğüne göre text dosyası, bir access dosyası veya bir SQL Server, Oracle bu işlevi yerine getiriyor olabilir. Yine verilerin büyüklüğüne göre; tek tablo yapısından, tablo tanımlamalarından, tabloların arasındaki ilişkileri ve kullanıcı tanımlarına destek veren view, stored procedure, kullanıcı tanımlı fonksiyon, trigger gibi yapılardan oluşur. Temel görevi veriyi, hızlı erişim için düzgün bir şekilde ve güvenli olarak saklamaktır. Dışarıdan söyleneni anlayabilmesi için SQL denilen (Structred Query Language-Yapısal Sorgu Dili) veya XML gibi bir veri ve yapılmak istenenler ile ilgili tanımları yapabilecek dile ihtiyaç duyar.

Bunun üstüne İş katmanı denir. ASP uygulaması için Visual Basic Script gibi bir dil bilinmesi gerekmektedir. Bu, tek başına yeterli değildir. Bunun yanı sıra bir de veritabanına erişecek API (OLE-DB, DB-library, ODBC gibi)ve veritabanından gelen verileri programın içerisinde yönetmeyi sağlayacak nesnelere (Conneciton, RecordSet, Data Adapter, DataSet kısaca ADO veya ADO.NET gibi.) ihtiyaç duyulur. Bunların dışında da bol bol kod yazılması gerekir. Çünkü bu katman asıl işin yapıldığı veri katmanıdır. İş katmanı veri katmanına genellikle bir API (Application Protocol Interface) üstünden ve SQL kullanarak erişir.

İş katmanı, kullanıcı arayüzü ile veri katmanı arasında veri alış-verişini düzenler ve iş kurallarının uygulanmasını sağlar. ASP, ASP.NET, Muhasebe programı vb. gibi Delphi, VB, Java, C ailesinden bir dil ile yazılmış bütün veritabanına erişen programlar bu gruba girer. Temel görevi, verinin üstündeki işlemleri denetlemek ve işin yerine getirilmesi için gerekli mantıksal kuralları sağlamaktır. Genel olarak, programlama ile kastedilen kısımdır.

Bundan sonra ASP uygulamasının ürettiği HTML kodlarının bir kullanıcının ekranında gösterilmesi işlemi gerekir. Bu işlemi gerçekleştiren üst katmana sunum katmanı denir. Sunum katmanı kullanıcı arayüzünden ibarettir. Örneğin, öğrenci isminde ve soyadında aranacak kelimenin girilebileceği bir kullanıcı arayüzü HTML kullanılarak yapılabilir.

### 1.3 RESULTSET Kavramı

VTYS'de bir sorgu çalıştırıldığında, tablo mantığında bir sonuç üretir. Bu sonuca resultset denir. Bir ResultSet, birden fazla tablodan kayıt içerebilir. Bir ResultSet'in içeriği, veritabanı programları geliştirilirken, ADO nesnelerinden RecordSet içerisine aktarılır ve veriler program içerisinde bu nesne aracılığıyla yönetilir. ADO.NET içerisinde ise ResultSet'ler (birden fazla resultSet) ADO.NET içerisinde ise DataSet denilen nesnelere aktarılabilir.

Resultset 'ler daha çok programlama dili içerisinde SQL kullanılarak elde edilen sonuçlardır ve daha sonrada veri tabanındaki bilgilerin programlama dili içerisinde kullanılması görevini yaparlar.

## 2. SQL PROGRAMLAMA, VERİ TİPLERİ ve DEĞİŞKENLER

### 2.1. SQL PROGRAMLAMANNIN AVANTAJLARI

SQL'de kullanılamayan procedür, fonksiyon, şartlı ifadeler, döngüler ve karşılaştırmaların eksikliğini giderebilmek için; Oracle PL/SQL (Programming Language/SQL), MS SQLServer ve Sybase T-SQL (Transact SQL) dilinin geliştirilmiş olduğu daha önce belirtilmişti. Oracle tarafından kullanılan PL/SQL komutları ile T-SQL bir çok noktada hemen hemen aynıdır ve SQL'de kullanılamayan procedür, fonksiyon, şartlı ifadeler, döngüler ve karşılaştırmalar kullanılabilir, PL/SQL ve T-SQL if,case,for..next gibi programlama için gereken işlemleri kullanmayı olanaklı kılar.

Birbirinden çok az farklarla ayrılan PL/SQL ve T-SQL 'e ikisini de kapsayacak şekilde **SQL programlama** denilir. Bu ders notu kapsamında PL/SQL ve T-SQL için SQL programlama kavramı kullanılmıştır. SQL programlama; veri üretimi ve SQL sorgulamalarının blok yapılar ve prosedür halinde kullanılmasını sağlayan güçlü bir bağımlı dildir. Sağladığı avantajlar aşağıda sıralanmıştır.

- SQL 'de olmayan prosedürel işlemler SQL programlama ile yapılabilir.
- Program geliştirmeyi modülerize eder.
- Mantıksal ilişkideki ifadeleri bir blok halinde kullanır.
- Karmaşık bir problemi daha anlaşılabilir, iyi tanımlanmış alt bloklar halinde parçalar.
- Değişkenleri, sabitleri, kursor ve dışlamaları (exceptions) tanımlar ve SQL içinde prosedürel olarak kullanır.
- Basit ve karmaşık veri tiplerine değişken atayabilir.
- Değişkenleri, veri yapısı üzerine ve veritabanındaki sütunlara atayabilir.
- Ardışık ifadeleri sırasıyla veya döngü içinde çalıştırır.
- Bir kursor yardımıyla birkaç satırlık sorgulama sonucu dönen satırı tek tek işler.
- Veri tabanı programlarının; TOOLS adı verilen veritabanı üzerinde programlama gerçekleştiren programları vardır ve bunlar SQL ve SQL programlama kullanırlar. Örneğin; Oracle Forms Oracle Veri tabanı üzerinde formlar oluşturur. Reports ise raporlar oluşturmayı sağlar. Tools programları kod yazım aşamasında SQL ve SQL Programlama kullanırlar. Kontrol için bir aradaki komutları gruplamak için SQL programlama veri tabanı araçları (tools) ile birleştirir. Veri tabanı geliştirme araçları (tools) için merkezi bir rol oynar.

- SQL programlama kullandığı veri tabanı ile uyumlu olduğundan, programları başka bir veri tabanı ve SQL programlama sağlayan herhangi bir ortama taşınabilir.
- SQL programlama birkaç SQL ifadesini birleştirir ve server'a bir defada gönderdiği için network trafiğini azaltır ve daha hızlı çalışmayı sağlar.

## 2.2 VERİ TİPLERİ (DATA TYPE) VE DEĞİŞKENLER

Bilgisayar, kayıtları tablolarda yapısal olarak tutarken, onların yapıları hakkında fikir sahibi olabilmek için bazı özelliklerinin önceden tanımlanması gerekir. Örneğin, personel sicil numarası alanının mutlaka bir tam sayıdan oluşacağı, personel ad ve soyadının harflerden oluşacağı, personelin çalıştığı bölümün harf ya da rakamlardan oluşacağı, personelin doğum tarihinin tarih bilgilerinden oluşacağı gibi. Bir veritabanı oluşturulurken, önce tablolar ve sonrada bu tablodaki her bir alanın veri tiplerinin ne olacağı tanımlanmak zorundadır. Bir tablo alanına veri girişi yapılmadan önce o alanın tamsayı mı yoksa harf mi; tarih mi yoksa ondalıklı bir sayı mı olacağı tanımlanmalı ve veriler daha sonra tabloya yazılmalıdır. Ayrıca, “bir alanın uzunluğu ne kadar olacak, harf girilebiliyorsa en fazla kaç harf girilebilecek?”, “rakam ise en fazla kaç basamaklı olabilir?” türünden soruları yanıtlamak için de yine VTYS bir alan için veri tipi belirlenmesini ister.

**Değişkenler;** programlama dillerinde, değeri daha sonra akışlara göre değişecek bir değer için hafızada (RAM'de) bir yer ayırmak için kullanılan yapıdır. Hafızadaki bu değere daha sonra değişkenin adı ile erişilir ve yeni değeri atanır. Değişkene atanan her yeni değer bir eski değeri siler.

SQL programlamada kullanılacak olan değişkenler veritabanı veri tiplerinden biri olmalıdır. Örneğin CHAR, DATE ve NUMBER veya PL/SQL veri tiplerinden biri olan BOOLEAN ve BINARY INTEGER gibi. Aşağıda örnek olması açısından Oracle veri tabanı veri tipleri listelenmiştir.

### 2.1.1. ORACLE VERİ TABANI VERİ TİPLERİ

**CHAR(sayı)** : Sabit uzunluktaki alfasayısal verilerin tutulabildiği alanlar için kullanılır. Oracle 7 ve daha önceki sürümler için bu alanın uzunluğu en fazla 255 karakter olabilir. Oracle 8 ve sonrasında 2000 karakter uzunluğundadır. Eğer, sayı ile ifade edilen numaradan daha kısa uzunlukta veriler girilirse Oracle kaydın sonuna boşluk ekleyerek sabit uzunluğa kadar getirir. Uzunluk sabit olarak belirlenir. Örnek char(20).

**VARCHAR2(sayı)** : Değişken uzunluklu alfasayısal verilerin tutulduğu alanlar için kullanılır. Oracle 7 ve önceki sürümlerinde 2000 karakter, Oracle 8 ve sonraki sürümlerinde 4000 karakter uzunluğunda bilgi girilebilir. Karakter türündeki veriler gibidir. Tek fark uzunluk değişkendir. Yani verinin ihtiyaç duyduğu uzunluk kullanılır. Örnek varchar2(30).

**NUMBER(tam,ondalıklı)** : Sabit veya değişken kesirli sayılar için kullanılan sayısal veri tipidir. Tam kısım en fazla 38 basamak olabilir. Ondalık kısmın basamak sayısı da -84 ile 127 arasında değişmektedir. Number veri tipinden türetilmiş int[eger], dec[imal], smallint ve real veri tipleri de kullanılabilir.

**LONG** : 2 GB 'a kadar bilgi tutabilen karakter alanlar için kullanılır. Bir tabloda bu tipten ancak bir adet alan tanımlanabilir. Long veri tipine sahip alanlar için index oluşturulamaz.

**BINARY-INTEGER** : İşaretsiz integer sayılarını saklamak için kullanılır. 2147483647 ile 2147483647 arasındaki sayılar için kullanılır.

**DATE** : Tarih tutan alanlar için kullanılır. Bu tip alanlarda, tarih bilgileri ve saat bilgileri tutulabilir. Tarih formatları Oracle yüklenilirken seçilen dile göre değişir. Amerikan standartı için 'DD-MON-YY' dir. Yani bir tarih '03-MAY-01' şeklinde görünür. NLS\_DATE\_FORMAT parametresi ile tarih formatı değiştirilebilir. Tarihsel alanlar üzerinde aritmetiksel işlemler yapılabilir. Sistem tarihi SYSDATE fonksiyonu kullanılarak öğrenilebilir. Sayısal veya karakter olarak tanımlı bir alandaki veriler TO\_DATE fonksiyonu ile tarih tipine çevrilebilir.

**BOOLEAN** : Mantıksal ifadeleri saklamak için kullanılır. Boolean tipte bir değişken "true" (doğru), "false" (yanlış) ve "null" değerlerini alabilir.

**Not:** Bir tablonun alanları kendi veri tipine uygun değerler alabildiği gibi bir de NULL değer alabilirler. NULL değeri sayısal olarak 0'dan ve karakter olarak ta boş karakterden(' ') farklıdır.

## 2.3 VERİ TABANI PROGRAMLARINDA VE SQL DE DEĞİŞKEN TANIMLAMA

SQL 'de tablo adları, alan (field), veritabanı dosyası, indeks vb. isimler değişken isimleridir. Genel değişken isimlendirme kuralları burada da geçerlidir.

1. Değişken isimleri, harf ile başlamak zorundadır.
2. Değişken isimleri, harf, rakamlar ve '\_' dan oluşmak zorundadır.
3. Değişken isimlerinde Türkçe noktalı harflerin (İ,ı,Ğ,ğ,Ü,ü,Ş,ş,Ç,ç,Ö,ö) ilerde dil problemi yaşanmaması açısından kullanılmaması gerekir.
4. Komut olarak ayrılmış kelimeler değişken adı olamazlar (select, like, not, or, delete, update vs.)
5. SQL büyük-küçük harf duyarlı değildir.
6. Değişken isimlerinde boşluk yer alamaz.

### Değişken isimlendirme notasyonları:

1. **Deve notasyonu** : degiskenAdı şeklinde yazılır.
2. **Alt çizgi notasyonu** : degisken\_adı şeklinde yazılır.

Veritabanı programlamada (SQL programlama), büyük-küçük harf duyarlılığı olmadığından genellikle alt çizgi notasyonu kullanılır ve değişken adları küçük harf olarak verilir. Ancak bu bir kural olmayıp sadece okunurluğu artırmak için programcıların bir çoğu tarafından tercih edilen bir yoldur.

**NULL mu, boşluk mu?** Bir kayıt için, alanlardan biri hiç girilmediği için boş olabilir veya bilgisayardaki space tuşunun karşılığı ASCII değeri girilmiş olabilir. Space (ASCII-32 karakteri) tuşuna basılarak elde edilmiş boşluk ile daha hiçbir bilgi girilmemiş olan boşluk bilgisayar dilinde birbirinden farklıdır. Daha önce hiçbir şey girilmemiş alan için NULL terimi kullanılır.

## 2.4 DEĞİŞKENLERE DEĞER ATAMAK

SQL programlamada DECLARE kısmı diğer programlama dillerinde olduğu gibi değişken ve değişkenlerin tanımlandığı kısımdır. Değişkenlere iki yöntemle değer atanabilir.

1. Burada tanımlanan değişkenlere “:=” operatörü ile ilk değer atanabilir. Eğer bir ilk değer atanmazsa değişkenlerin alacağı ilk değer “null” ‘dur. Eğer “constant” ile tanımlama yapılırsa değişkenin değeri değiştirilemez.

```
DECLARE
isbatar          date;          /* ilk değeri “null” */
isi              varchar2(80) := 'tezgahtar';
isci_bulundu     boolean;        /* ilk değeri “null” */
maas_artisi      constant number(3,2) := 1.5; /* sabit */
BEGIN ... END;
```

Tanımlama kısmında değişkenlere bir veri tipi vermek yerine bir tablodaki bir alanın veri tipi değişkene aktarılabilir. %TYPE niteliği, değişkenlerin, sabitlerin veya veritabanı sütunlarının (alanlarının) veri tiplerini elde eder. Bu özellikte veritabanı sütunlarını değişkenlere referans etmek için kullanılır. Bu özelliği kullanmak şu faydaları sağlar: Değişkenler veri tabanı sütunlarına referans edilmiş olduğundan; veri tiplerini her zaman bilmek gerekmez veya veritabanı sütunlarının veri tipleri değiştirildiği zaman bir problem oluşmaz.

Örneğin “isci.iscino%TYPE” şeklinde bir tanımlama ilgili değişkenin “isci” tablosundaki “iscino” değişkeni ile aynı veri tipinde olmasını sağlar.

```
DECLARE
V_soyadı         t_isci.soyadı % TYPE;
V_adı            t_isci.adı % TYPE;
V_maas           NUMBER(7,2);
V_min_maas       V_maas % TYPE := 10;
```

Tek bir değişken için tablonun bir alanının veri tipini almak yerine tablonun tüm alanlarının veri tipleri bir değişkene aktarılabilir. Örneğin “isci%ROWTYPE” ile istenen değişkene tablonun yapısı aynen aktarılabilir. Buna tıpkı Pascal’daki gibi “record” veri tipi denir. Burada eğer “tablo isci%ROWTYPE” şeklinde bir tanımlama yapılırsa tablo.iscino ya da tablo.isciadi şeklinde değişkenler kullanılabilir. Bu tür bir tanımlama imleç kullanılırken kolaylık sağlar.

```
DECLARE
    Emp_rec emp % ROWTYPE;
    CURSOR c1 IS SELECT deptno, dname, loc FROM dept;
    Dept_rec c1 % ROWTYPE;
```

İlk satır işçi tablosundan seçilen bir satırdır. İkincisi C1 kursorü ile fetch edilen bir satırdır.

**Örnekler :**

```
ax := price * tax_rate;
bonus := current_salary * 0.10;
raise := TO_NUMBER(SUBSTR('750 raise',1,3));
valid := FALSE;
```

**2. Select veya FETCH veritabanı değerlerini atama işlemini yapar. Aşağıda örnek işçi nosu işçi\_no'ya eşit olan işçinin ikramiyesine maaşının %0.10 atanır.**

```
SELECT maas * 0.10 INTO ikramiye
FROM isci WHERE isno=isci_no;
```

**Not 1 :** SQL programlamada bir çok diğer programlama dillerinin aksine **T-SQL** 'deki değişkenler @ ile başlamak zorundadır. Her komut sonuna da GO kelimesi eklenmelidir. PL/SQL'de her satırın sonuna noktalı virgül işareti konulması gerekirken T-SQL'de bu kural geçerli değildir.

```
DECLARE @kitapNo INT
DECLARE @kitapAdi VARCHAR(63)
```

Yada bir satırda birden fazla değişken tanımlayabiliriz.

```
DECLARE @kitapNo INT, @kitapAdi VARCHAR(63)
```

**Örnek :**

```
DECLARE @enSonEklenenKitap INT
SELECT @enSonEklenenKitap=MAX(kitapNo) FROM kitap
```

**Not 2 :** SQL programlamada bazı satırlar açıklama satırı olarak kullanılmak istenebilir. Bu durumda satırın açıklama satırı olduğu "/" ile başlayan ve "/" ile biten karakterler arasına yazılır. T-SQL de bir satırın dikkate alınmaması isteniyorsa "- -" iki çizgi ile bu belirtilebilir. MSSql Server ve Sybase "- -" işaretini bazı tool 'larında dikkate almazlar. Oracle veritabanında "/" açıklamalar "/" şeklinde kullanılır.



## 3. SQL PROGRAMLAMA BLOKLARI ve AKIŞ DENETİMİ

### 3.1 SQL PROGRAMLAMA BLOKLARININ YAPISI

Birbirinden çok az farklarla ayrılan PL/SQL ve T-SQL 'e ikisini de kapsayacak şekilde **SQL programlama** demiştik ve bu ders notu kapsamında PL/SQL ve T-SQL için SQL programlama kavramı kullanmıştık. Aralarında olan farkları belirtmek açısından sizlerinde çalışma ortamında yanlışlık yapmamanız için ders notlarının bölümlerinde bazen PL/SQL bazen T-SQL yapıları ayrı ayrı incelenebilecektir. Buna dikkat ederek, aradaki farklılıkları görüp kullandığınız SQL programlamasına göre kullanımını uygulayınız.

PL/SQL ve T-SQL blok yapılı bir dildir. Her bir blok bir program ünitesini oluşturur. PL/SQL blokları prosedür, fonksiyon ve normal blok olmak üzere üçe ayrılır. Prosedür ve fonksiyon yapısı ilerleyen bölümlerde ayrıca incelenecektir. Bu bölümde normal bir blok yapısı ve akış denetimi anlatılmıştır. Bir PL/SQL bloğu seçimsel bir tanımlama bölümü, PL/SQL cümlelerinin yazıldığı bir bölüm ve hata yakalama bölümünden oluşur.

Bloklara bir isim verilebilir. İsim vermek zorunlu olmasa da kullanımda kolaylık sağlaması açısından tavsiye edilir. Eğer hazırlanan bir blok yeniden kullanılmak istenirse .sql uzantılı dosyalara saklanarak yeniden kullanılabilir. Bir PL/SQL blok yapısı aşağıdaki gibidir :

```
[<blok başlığı>]
[DECLARE
<sabitler>
<değişkenler>
<imleçler>
<kullanıcı tanımlı hata yakalama isimleri>]
BEGIN
<PL/SQL komutları>
[ EXCEPTION
<hata durumu komutları>]
END;
```

Yukarıda bir PL/SQL blok yapısının genel kullanımı yazılmıştır. Burada \* “[” ve “]” işaretleri arasındaki alanların yazılması zorunlu değildir.

Blok başlığı PL/SQL bloğunun prosedür, fonksiyon veya bir paket bloğu olup olmadığını belirler. Eğer bir başlık tanımlanmazsa bu isimsiz blok (anonymous) olarak adlandırılır. “Declare” kısmı diğer programlama dillerinde olduğu gibi değişken ve sabitlerin tanımlandığı kısımdır.

PL/SQL blokları içerisinde kullanılan tüm sabitler, değişkenler, imleçler ve kullanıcı tanımlı hata durumları “declare” kısmında tanımlanmalıdır. Burada sabit ve değişkenler şöyle tanımlanabilir:

<değişken adı> [constant] <veri tipi> [not null] [:= <ilk değer>];

Oracle ‘da kullanılan tüm veri tipleri daha önce anlatılmıştı.

PL/SQL blokları içerisinde gerçekleştirilecek işlemler begin..end kelimeleri arasına yazılır. Burada dikkat edilmesi gereken bir husus vardır: PL/SQL blokları içerisinde veri tanımlama dili komutları(yani Create Table, Alter TableSpace, Drop User gibi) ve veri kontrol dili komutları(grant ve revoke gibi) kullanılamaz. PL/SQL blokları içerisinde veri işleme dili komutları kullanılabilir. (Select, Update, Delete, Insert gibi) .

Hata durumları ya da aykırı durumlar olarak adlandırılan “exceptions” kısmında PL/SQL blokları arasında gerçekleşen bazı hataları kullanıcıya yansıtmadan kontrol etme ve gerekli işlemleri yapma olanağı vardır. Oracle tarafından tanımlanmış hata durumları olduğu gibi programcılar da hata durumları tanımlayabilirler.

## 3.2 SQL PROGRAMLAMADA AKIŞ KONTROLLERİ

### 3.1.1. PL/SQL AKIŞ KONTROLLERİ

PL/SQL blokları içerisinde koşullu-koşulsuz dallanmalar ve döngüler kullanılabilir. PL/SQL’de iki tip kontrol yapısı vardır. Birincisi “IF” kontrol yapısı ve diğeri de “LOOP “ kontrol yapısıdır. ELSEIF ifadesi birleşik END IF ifadesi ayrı yazılır

**A- IF Kontrol Yapısı :** Üç tip “IF” yapısı kullanılır:

1-) IF şart THEN Komutlar END IF	2-) IF şart THEN komutlar ELSE Şartın gerçekleşmemesi halindeki komutlar END IF	3-) IF şart THEN komutlar ELSEIF şart komutlar END IF
--	--	---

1.yapıda şart kontrol edilir ve doğru ise THEN deyiminden sonra yazılan komutlar işlem görür. Şart gerçekleşmiyorsa program ENDIF deyiminden sonraki satıra geçer ve işlemlere devam eder.

2.yapıda şart kontrol edilir ve doğru ise THEN deyiminden sonra yazılan komutlar işlem görür ve program ENDIF deyiminden sonraki satıra geçer ve işlemlere devam eder. Şart gerçekleşmiyorsa ELSE deyiminden sonra yazılan komutlar işlem görür ve program ENDIF deyiminden sonraki satıra geçer ve işlemlere devam eder.

3.yapıda şart kontrol edilir ve doğru ise THEN deyiminden sonra yazılan komutlar işlem görür ve program ENDIF deyiminden sonraki satıra geçer ve işlemlere devam eder. Şart gerçekleşmiyorsa ELSEIF deyimindeki şart kontrol edilir. Buradaki şart doğru ise ELSEIF satırından sonraki komutlar işlem görür. Burada ELSEIF kısmına istenilen kadar ELSEIF şartı eklenebilir. Program her seferinde ELSEIF şartını kontrol eder, doğru ise komutları yapar değilse bir sonraki şarta geçer ve en sonunda da ENDIF den sonraki satırlara geçerek işlemlere devam eder.

İlk IF deyimi sonucu; hareketlerinin her iki kümesi gerçekleştirilen belirli hareketlerden önce daha ileri IF deyimlerini içerir. Her iç içe IF deyimi karşılığı olan bir END IF ile bitirilmelidir. Mümkün olduğunda ELSEIF cümlecisi iç içe IF deyimlerinin yerine kullanılır. Kod okumak ve anlamak için kolaydır. Mantık kolayca teşhis edilir. Eğer ELSE cümlecisindeki hareket tamamıyla diğer IF deyimini kapsıyorsa, ELSEIF cümlecisini kullanmak daha uygundur. Bu, daha ilerideki koşul ve hareket kümelerinin her birinin sonundaki iç içe END IF lerin ihtiyacı olan kodların daha net anlaşılmasını sağlar.

**Örnek :** Soyadı Kılıç olan müşterinin meslek alanına satış temsilcisi ve bölge numarasına 35 yazan PL/SQL kod parçası.

```
....
IF m_soyad = 'Kılıç' THEN
    M_gorev = 'Satış Temsilcisi';
    M_bolgeno = 35;
END IF;
.....
```

**Örnek :** Fatura düzenlenme tarihi ile gönderme tarihi arasında 5 günden az olan faturalara 'Kabul' diğerlerine 'Red' yazan PL/SQL kod parçası.

```
....
IF f_duzenleme_tarihi - f_gonderme_tarihi < 5 THEN
    F_onay = 'Kabul';
ELSE
    F_onay = 'Red';
END IF;
.....
```

**B- DÖNGÜ Kontrol Yapısı :** LOOP, FOR ve WHILE döngüleri kullanılır. :

**1- LOOP :** En basit döngü tekrarlatmak için sınırlayıcılar olan LOOP ve END LOOP arasında çevrelenmiş ifadelerin gövdesini içerir. Her seferinde programın akışı END LOOP deyimine ulaşır, denetim END LOOP ifadesinin altında karşılığı olan LOOP deyimine geri döndürülür. Bu denetlenmemiş döngü göz ardı edilen bir döngüdür. Bu sonsuz döngüden kurtulmak için EXIT ifadesi eklenmelidir.

Bir döngü EXIT ifadesi kullanılarak bitirilebilir. END LOOP deyiminden sonra denetim bir sonraki ifadeye geçer. Hem bir IF deyimindeki bir hareketi olarak hem de döngü içindeki bağımsız bir deyim olarak EXIT yayınlanabilir. Diğer bir durumda bir WHEN cümlecisi döngünün koşulu bitirmesine müsaade etmek için bağlanabilir.

```
LOOP
    Komut1
    .....
    KomutN
    GOTO etiket adı
EXIT [WHEN şart]
END LOOP
```

**örnek :**

```
.....
v_ord_id s_item.ord_id%TYPE:=101;
v_counter NUMBER(2) :=1;
BEGIN
.....
    LOOP
        INSERT INTO s_item (ord_id, item_id)
        VALUES (v_ord_id, v_counter);
        V_counter := v_counter + 1;
        EXIT WHEN v_counter > 10;
    END LOOP
.....
```

**1- FOR Döngüsü :** Diğer programlama dillerindeki FOR ile benzerlik gösterir. Ayrıca, ek olarak PL/SQL 'in gerçekleştirdiği yinelemelerin numarasını bitirmek için FOR döngüleri LOOP anahtar kelimesinin önüne bir denetim deyimine sahiptir.

```
FOR sayac IN [REVERSE] başlangıç..bitiş
LOOP
    Komut1;
    ...
    KomutN;
EXIT [WHEN şart]
END LOOP;
```

**3- WHILE Döngüsü :** Bu döngü denetim koşulu TRUE olmayana kadar deyimlerin bir serisini tekrarlamak için kullanılabilir. Her yinelemenin başlangıcında koşul hesaplanır. Koşul FALSE (yanlış) olduğu zaman döngü sona erer. Eğer koşul döngü başlangıcında FALSE (yanlış) ise daha ileri olmayan yinelemeler gerçekleştirilir.

```
WHILE şart LOOP
    Komut1;
    ...
    KomutN;
END LOOP;
```

Eğer koşul içindeki şartta belirtilen ilgili değişkenlerin değerleri; döngü içerisinde değiştirilmezlerse kod parçasında koşul TRUE kalacağından döngü sonsuz olur ve sona ermez.

### 3.1.2. T-SQL AKIŞ KONTROLLERİ

T-SQL 'in sunduğu belli başlı akış kontrolleri şunlardır :

#### A- IF ... ELSE Karar yapısı

If (şart1) deki şart1 doğru ise if (şart1) 'den sonra gelen BEGIN ile END arasındaki kodlar çalıştırılır. Daha sonra If bloğunun en altındaki kodlardan devam eder. Genel yapısı şu şekildedir.

```
If (şartlar)
    begin
    ...
    end
Else if (şartlar)
    Begin
    ...
    end
else
    begin
    ....
    End
```

Şart1 doğru değil ise, elseif (şart2) deki şart2'ye bakılır. Bu şart doğru ise, şartı takip eden BEGIN ile END ifadeleri arasındaki kodlar çalıştırılır, if bloğunun sonuna gider ve takip eden kodları çalıştırır. Şart2 'de yanlışsa, elseif (şart3) deki şart3 'e bakılır. Şart3 doğru ise takip eden BEGIN ile END deyimleri arasındaki kodlar çalıştırılır, if bloğunun en sonuna gider ve takip eden kodları çalıştırır. Şart3 yanlış ise bu aradaki kodları da çalıştırmadan geçer.

**Örnek :** Bir işyerindeki en yüksek maaş alan personel hakkında bilgi veren T-SQL program kodu :

```
Declare @ymaas int
Select @ymaas=max(maas) from maaslar
If (@ymaas>2000)
    Begin
        Print 'bu işyerinde iyi maaş alan biri var'
    End
Else
    Begin
        Print 'bu işyerinde iyi maaş alan biri yok'
    End
```

If yapısının en yaygın kullanımlarından birisi de IF EXISTS yapısıdır. Genellikle bir veri yada nesnenin daha önceden var olup olmadığı bu yapı ile test edilir.

```
IF NOT EXISTS (select * from personel)
DROP TABLE personel
```

Örnekte Personel tablosunda hiçbir kayıt yoksa tablo silinmiştir.

## B- CASE Deyimi

Bir karar döngüsüdür ve oldukça kısa kod ile etkin programlar oluşturmayı sağlar.

```
CASE
WHEN şart THEN değer
[ELSE değer]
END
```

Örneğin bir kütüphanede ödünç verilen kitaplar için 'dışarıda' ödünç verilmeyip kütüphanede kalanlar içinde 'içeride' yazan bir T-SQL program kodu yazılabilir.

	kitapNo	geldiMi	kitapDurumu
1	1	1	Içeride
2	5	1	Içeride
3	1	1	Içeride
4	6	1	Içeride
5	1	0	Dışarıda
6	2	0	Dışarıda
7	3	0	Dışarıda

```
SELECT kitapNo, geldiMi, 'kitapDurumu'=
CASE
    WHEN geldiMi=0 THEN 'Dışarıda'
    WHEN geldiMi=1 THEN 'içeride'
END
FROM odunc
```

## C- WHILE Döngüsü

Tekrar gerektiren işlemlerde istenilen şart gerçekleşinceye kadar işlem yapmaya olanak sağlar. While ile bir işlemi istenilen kadar tekrarlatılabilir. Genel yapısı şu şekildedir:

```
WHILE şart
BEGIN
    Tekrarlanması gereken kodlar
END
```

Şart gerçekleşinceye kadar, BEGIN ile END arasına yazılan kodlar işlem görür.

**Örnek :** Integer tanımlanan bir değişkeni değeri 15 olana kadar 1'er artırıp, 15 olduğunda yazan T-SQL kod parçası.

```
DECLARE @sayac INT
Select @sayac=1
WHILE (@sayac<15)
BEGIN
    SELECT @sayac=@sayac+1
END
Print 'sayac : '
Print @sayac
```

**Not :** WHILE döngüsünde, döngünün devam edip etmeyeceğini tayin eden şart kısmına, mutlaka sağlanacak bir koşul yazılmalıdır. Aksi halde, sonsuza kadar çalışacak bir kod yazılmış olur. Buda SQL Server'ın kilitlenmesine neden olur. Aşağıda BREAK komutu ile birlikte bu konu ile ilgili bir örnek bulunmaktadır.

**BREAK Komutu :** WHILE döngüsünden bir uç şarttan dolayı çıkmak için kullanılır. BREAK ile while döngüsünden çıkıldığında, WHILE'in END'ini takip eden kodlardan devam edilir.

**Örnek :** Integer tanımlanan bir değişkeni değeri 15 olana kadar 2'ser artırıp, 15 olduğunda döngüden çıkan T-SQL kod parçası.

```
DECLARE @sayac INT
Select @sayac=0
WHILE (@sayac<>15)
    BEGIN
        SELECT @sayac=@sayac+2
        If @sayac > 15
            BEGIN
                Print 'bir alt satır olmasa kısır döngü hatası olacaktı'
                Break
            END
        ELSE
            BEGIN
                Print 'henüz bir kısır döngü hatası olmadı'
                Break
            END
    END
Print 'sayac : '
Print @sayac
```

Bu örnekte, @sayac değişkeni 14 ve 16 değerlerini alır ama 15 değerini almaz. Eğer durum if deyimi ile kontrol edilmeseydi ve break komutu olmasaydı program sonsuza kadar çalışacak bir döngüye girerdi ve SQL-Server kilitlenirdi.

**CONTINUE Komutu :** Break komutu gibi, WHILE döngüsünde bir uç şartı kontrol etmek için kullanılır. Ancak bu komut, break'ın aksine WHILE yapısının başlangıcına götürür ve eldeki değerlerle oradan programcının devam etmesini sağlar.

**Örnek :** 1 den 15'e kadar 11 hariç tüm sayıları yazdıran T-SQL kod parçası

```
DECLARE @sayac INT
SELECT @sayac=1
WHILE (@sayac < 15)
    BEGIN
        SELECT @sayac=@sayac+1
        If (@sayac=11)
            BEGIN
                Continue
            END
    END
Print 'sayac : '
Print @sayac
END
```



Sonuçta, 1,2,3,4,5,6,7,8,9,10,12,13,14,15 yazdırılacak ama 11 yazılmayacaktır.

### 3.3 SQL GEÇİCİ TABLOLARI

Bazen geçici bir süre için ek tablolara ihtiyaç duyulabilir. Bu tür durumlarda, T-SQL ile geçici tablolar oluşturulup onlar kullanılabilir. Geçici tablolar, kullanıcı çıkış yaptığı veya SQL Server kapatıldığında otomatik olarak silinirler. Geçici tablo oluşturmanın iki yöntemi vardır.

**1. Yöntem :** Oturum boyunca geçerli geçici tablolar oluşturmak için kullanılır.

```
CREATE TABLE #tablo_adı ( ..... )
```

Tablo\_adı alanının başına # işareti konur ve normal create table komutu aynen yazılır. Bu şekilde oluşturulan tablolar oturum kapandığında veya SQL Server durdurulduğunda silinir. İstenirse tablo bilinen yöntemle de silinebilir.

```
DROP TABLE #kitap
```

**Örnek :** Geçici olarak Kitaplar tablosunun yaratılması:

```
CREATE TABLE #kitap (  
    KitapNo INT,  
    KitapAdi VARCHAR(55),  
    ISBNNo CHAR(16)  
)
```

Daha sonra tablo üzerinde normal bir tabloymuş gibi işlemler yapılabilir. Kayıt güncellenebilir, silinebilir veya listelenebilir.

```
SELECT * FROM #kitap      veya  
INSERT INTO #kitap VALUES (.....) gibi.
```

**2. Yöntem :** Tempdb adlı veritabanı dosyasına bir tablo açılır. Bu veritabanındaki tablolar sadece SQL Server kapatıldığında silinir. Genel yapısı şöyledir :

```
CREATE TABLE tempdb..tablo_adı ( ..... )
```

Tempdb adlı veritabanı SQL server açıldığı anda başlayan SQL Server durduğu anda kaybolan bir tablo olduğu için oluşturulan geçici tablo da SQL Server kapatıldığında silinir.

## 4. SQL PROGRAMLAMADA PROCEDURE ve FUNCTION

### 4.1 PROCEDURE (PROSEDÜR) VE ÖZELLİKLERİ?

Prosedürler belirli işlemleri gerçekleştirmek üzere oluşturulan özel bloklardır. PL/SQL'de prosedürler diğer programlama dillerinden biraz farklı olarak birden fazla değer döndürebilirler.

Nesneye dayalı programlama bu kadar popüler değilken, programlar sadece prosedür denilen parçacıklardan oluşurdu. Prosedürler, belli bir işlevi yerine getirmek için özenle yapılandırılmış program parçacıklarıdır. Mesela, iki sayı alıp bunların toplamalarını hesaplayan bir kod parçası toplama adında bir prosedür içerisine paketlenabilir. Bir prosedür, başka bir prosedür içerisinden çağrılabilir. Bu da sık kullanılan işlemler için yazılmış kodların bir defa yazılıp çok defa kullanılmasını böylelikle de programlamayı kolaylaştırmayı amaçlar.

Prosedürler, bir çok gelişmiş programlama dilindeki fonksiyon yapılarına karşılık gelirler. SQL programlama ile yapılabilen birden fazla her işlemin, paketlenmiş bir halde bir tek komut ile çalıştırılması gerektiğinde procedure 'ler kullanılır.

T-SQL 'de Stored procedure (SP), 1980'li yılların sonunda Sybase SQLServer ile birlikte kullanıma girmiştir. En büyük özelliği sorguların önceden hazırlanması, derlenmesi ve VTYS ile aynı uzayda çalışmasından dolayı daha hızlı sonuç vermesidir.

Bir SP oluşturulduktan sonra, veritabanı sunucusunda saklanır. Her ihtiyaç duyulduğunda aynı SP defalarca çağrılabilir. Cursor gibi oturum kapandığında silinmez.

SP ler Network bazlı çalışmalarda ağ trafiği ve sistem kaynaklarının kullanımını düzenleyerek de performans artışı sağlarlar. Bir dize işlem, bir tek paket içerisinde yer alır. Gerektiğinde bir tek komut ile tetiklenebilir. Paketin tamamı çalışıncaya kadar istemde bulunan terminale hiçbir şey gönderilmez. Tüm komutlar bittiğinde bir tek sonuç gönderilir. Bu da bazı durumlarda ağ trafiğini rahatlatır.

#### **Bir Procedür sistem tarafından oluşturulduğu anda şu aşamalara tabi tutulur:**

1. Procedür bileşenleri parçalara ayrıştırılır
2. Veritabanı içerisinde table, view gibi başka nesnelere atıfta bulunan referanslar varsa, geçerli olup olmadıkları kontrol edilir. (Geçerli:1-nesne var mı, 2-izin var mı)
3. Kontrollerden geçen Procedür'ün adı ve kodları ise sistem tablolarında saklanır.

4. Bu işlemlerle birlikte derleme işlemi yapılır. Normalizasyon işlemleri olarak da anılan bu işlemler sonucunda, ağaç şeması elde edilir.

5. Procedür herhangi bir anda çağrıldığında, ilk kez çalışıyorsa bu işlemler gerçekleştirilir. İlk sefa çağrılmıyorsa, kontrol, sorgulama ağacı oluşturma işlemleri yapılmaz ve oldukça hızlı bir şekilde Procedür'ün derlenmiş hali çalışır. Bundan dolayı prosedürler derlenen nesnelerden biri olarak anılır.

#### **Procedürler şu faydaları sağlar:**

1. Uygulamanın getirdiği bazı iş kuralları prosedür içinde tanımlanabilir. Bir kez oluştuktan sonra bu kurallar birden çok uygulama tarafından kullanılarak daha tutarlı bir veri yönetimi sağlanır. Ayrıca bir fonksiyonelliğin değişmesi ihtiyacı doğduğunda her uygulama için değişiklik yapmak yerine, sadece bir platformda değişiklik yapılır.

2. Tüm prosedürler üstün performansla çalışır ancak birden fazla çalıştırılacak olan prosedürler sorgulama planları procedure tamponcache içinde saklandığından daha da hızlı çalışırlar.

3. Procedure'ler veritabanı başlatıldıktan sonra otomatik olarak çalıştırılmak üzere ayarlanabilirler.

4. Procedure'ler harici olarak kullanılırlar. Trigger'lerden farklı olarak prosedürler uygulama tarafından ya da script tarafından bir şekilde çağrılmak zorundadırlar. Otomatik devreye giremezler.

5. Procedure'lerin içinde SQL sorgulama diline ek olarak SQL programlama komutları kullanılabilir.

6. Kullanıcının bir tabloya erişim izni olmasa bile o tablo üzerinde işlem yapan bir procedure'ü kullanma izni olabilir.

## **4.2 PROCEDURE (PROSEDÜR) OLUŞTURMA VE DEĞER DÖNDÜRME**

### **4.1.1. PL/SQL PROCEDÜRLERİ VE FONKSİYONLARI**

#### **A- PL/SQL Procedürleri :**

Prosedürler belirli işlemleri gerçekleştirmek üzere oluşturulan özel bloklardır. PL/SQL'de prosedürler diğer programlama dillerinden biraz farklı olarak birden fazla değer döndürebilirler.

```
PROCEDURE isim [(parametre[, parametre, ...])] IS
    [yerel tanımlamalar]
BEGIN
    komutlar
[EXCEPTION
    hata durumları]
END [isim];
```

Buradaki parametrenin yazılımı aşağıdaki gibidir.

```
parametre_adı [IN | OUT [NOCOPY] | IN OUT [NOCOPY]] veritipi  
[{: = | DEFAULT} açıklama]
```

Bir prosedürün iki kısmı vardır: tanımlama ve gövde. Tanımlama kısmı PROCEDURE kelimesi ile başlar ve prosedür adı ya da parametre listesi ile biter. Parametre tanımlamaları zorunlu değildir. Parametre kullanılmayan prosedürler parantez kullanılmadan yazılabilirler.

Prosedürün gövde kısmı IS anahtar kelimesi ile başlar ve END anahtar kelimesi ile biter. Prosedürün gövdesi de üç kısma ayrılır: değişkenlerin tanımlandığı kısım, komut cümlelerinin yazıldığı kısım ve hata durumlarının kontrol edildiği kısım. Değişken tanımlama kısmı IS kelimesinden hemen sonra başlar. Burada DECLARE kelimesi kullanılmaz. Komut cümlelerinin yazıldığı kısım ise BEGIN anahtar kelimesi ile başlar ve EXCEPTION ya da END ile biter. Bu kısımda en az bir komut yazılmalıdır. Hata durumları kısmı zorunlu değildir. Prosedür END kelimesi ile son bulur. Bu anahtar kelimenin yanına prosedür ismi yazılabilir, zorunlu değildir.

**Örnek :** Aşağıdaki maas\_artir prosedürü bir işçinin maaşının verilen miktar kadar artırılmasını sağlar:

```
PROCEDURE maas_artir (isci_no INTEGER, miktar REAL) IS  
    gecerli_ucret REAL;  
    ucret_yok EXCEPTION;  
BEGIN  
    SELECT ucret INTO gecerli_ucret FROM isci  
        WHERE iscino = isci_no;  
    IF gecerli_ucret IS NULL THEN  
        RAISE ucret_yok;  
    ELSE  
        UPDATE isci SET ucret = ucret + miktar  
            WHERE iscino = isci_no;  
    END IF;  
EXCEPTION  
    WHEN NO_DATA_FOUND THEN  
        print 'Yanlış İşçi Numarası'  
    WHEN ucret_yok THEN  
        print 'Geçerli Maaş Miktarı Yok';  
END maas_artir;
```

Bu prosedür bir başka blok içerisinde şöyle çağırılabilir:

```
DECLARE  
    isci_no NUMBER;  
    miktar REAL;  
BEGIN  
    ...  
    maas_artir(isci_no, miktar);
```

**B- PL/SQL Fonksiyonları :**

Bir fonksiyon bir değer hesaplayan alt programdır. Fonksiyon ve prosedür yapıları RETURN anahtar kelimesi haricinde benzerdir. Fonksiyonlar şöyle yazılabilirler:

```
FUNCTION isim [(parametre[, parametre, ...])] RETURN veri_tipi IS
    [yerel tanımlamalar]
BEGIN
    Komut cümleleri
[EXCEPTION
    Hata durumları]
END [isim];
```

Parametre listesi şu düzende verilebilir:

```
parametre_ismi [IN | OUT [NOCOPY] | IN OUT [NOCOPY]] veri_tipi
    [{:= | DEFAULT} açıklama]
```

Yukarıda prosedürler için anlatılan yapı kısmı fonksiyonlar içinde geçerlidir. Fark olarak burada fonksiyon tek bir değer döndürür ve bu değerın tipi RETURN anahtar kelimesinden sonra yazılır.

Prosedürler parametre listesi içerebilirler; fakat değer döndürmek zorunda değildirler. Bir fonksiyon iki yönüyle procedürden farklıdır.

- Fonksiyon bir ifadenin bir parçası olarak kullanılabilir.
- Fonksiyon değer döndürmek zorundadır.

Aşağıdaki fonksiyon bir maaş miktarının istenen dereceye ait olup olmadığını bulur:

```
FUNCTION maas_ok (maas REAL, ucret_derece INTEGER) RETURN BOOLEAN IS
    min_ucret REAL;
    max_ucret REAL;
BEGIN
    SELECT endusuc, enyukuc INTO min_ucret, max_ucret
    FROM ucretoran
    WHERE derece = ucret_derece;
    RETURN (maas >= min_ucret) AND (maas <= max_ucret);
END maas_ok;
```

Bu fonksiyon bir başka blok içerisinde şöyle çağrılabilir:

```
DECLARE
    yeni_ucret REAL;
    yeni_derece NUMBER(5);
BEGIN
    ...
    IF maas_ok(yeni_ucret, yeni_derece) THEN ...
```

Bir PL/SQL alt programı Oracle'ın tüm diğer ürünlerinde de kullanılmak istenirse bu alt program veritabanına kaydedilir. Alt programları Oracle veritabanına kalıcı olarak kaydetmek için CREATE PROCEDURE ve CREATE FUNCTION komutları kullanılır.

**Örnek :** işçi\_sil isimli procedür SQL ile yaratılmış ve daha sonraları da kullanılmak üzere kaydedilmiştir.

```
SQL> create procedure isci_sil(isci_no NUMBER) as
2  begin
3      delete from isci where iscino = isci_no;
4  end;
5  /

Yöntem yaratıldı.
```

Şekil 4.2.1. PL/SQL de Procedure Tanımlama Ekranı

### C- Procedure ve Function Parametre Modları

Fonksiyon ve prosedürler anlatılırken bunların parametreler geçirebileceği (döndürebileceği) söylenmişti. Procedure ve function ikisine birden alt program ismi verilir. Bir alt programa parametreler üç farklı modda geçirilebilir. Bunlar IN, OUT, IN OUT olarak adlandırılır. Bu üç mod her alt programda kullanılabilir. Fakat OUT ve IN OUT modlarının fonksiyonlarda kullanılmaması önerilir. Çünkü bir fonksiyonun amacı sıfır, bir ya da daha çok parametre alıp sonuçta tek bir değer göndermektir.

IN modunda alt programa geçirilen bir parametre bir sabit gibi davranır. Yani alt program içerisinde bu parametrenin değeri alınıp başka bir değişkene aktarılabilir, fakat değiştirilemez. Eğer parametre geçirirken hiçbir mod belirtilmezse, parametreler otomatik olarak IN modunda kabul edilir. IN modunda gönderilen parametre bir sabit, string, değişken ve matematiksel bir işlem olabilir.

OUT modunda gönderilen bir parametre alt program içerisinde bir değişken gibi davranır. Parametrenin değeri alınabilir ve değiştirilebilir. Bu tip bir parametre mutlaka bir değişken olmalıdır. Yani bir sabit ya da matematiksel bir işlem bu modda gönderilemez. Burada alt programa gönderilen parametrenin değeri, alt programa geçtiği anda NULL değer olur. Bu yüzden alt programa OUT modunda gönderilecek bir parametre tanımlanırken NOT NULL kısıtlamasının getirilmemiş olmasına dikkat edilmelidir. Alt program bittikten sonra, alt program içerisinde değer atanmış olan OUT modu parametreleri bu değerlerini korurlar. Yani girişte NULL değer alıp çıkışta değerlerini korurlar.

IN OUT modunda gönderilen parametreler diğer iki modun yaptığını birleştirirler. Yani bir parametre bir ilk değer ile alt programa girer, alt program içerisinde bu değeri değişebilir ve değerini kaybetmeden alt programdan çıkar. IN OUT modunda gönderilen bir parametre bir değişken olmalıdır, sabit, string ya da matematiksel bir işlem olamaz.

**Örnek :** Faktöriyel Hesabının procedür ve fonksiyon ile yapılması ve PL/SQL içersinden çağırılması aşağıdadır. Örneğe ve aradaki farklara dikkat ediniz.

Procedür ile yapılmış hali :

```
PROCEDURE faktoriyel_islemi (n Number) (n IN NUMBER, Sayi IN OUT NUMBER) IS
BEGIN
    FOR I IN 1...N LOOP
        Sayi = Sayi * I;
    END LOOP;
END;
```

Fonksiyon ile yapılmış hali :

```
FUNCTION faktoriyel_islemi (n Number) RETURN Number IS
BEGIN
    IF n=1 THEN
        RETURN 1;
    ELSE
        RETURN n * FAKTORİYAL (n-1);
    END IF;
END;
```

PL/SQL bloğu içersinden procedür ve fonksiyonun çağırılması ve çalıştırılması :

```
DECLARE
BEGIN
    /* procedürün çağırılması */
    sayi := 1;
    faktoriyel_islemi ( n, sayi);
    /* fonksiyonun çağırılması */
    sayi := faktoriyel_islemi (N);
END;
```

### **Bir Procedür veya Fonksiyonun Silinmesi;**

DROP PROC sp\_adi                      veya      DROP FUNCTION sp\_adi

yazılarak SP silinebilir.

**Örnek :** DROP PROC faktoriyel\_islemi                      veya      DROP FUNCTION faktoriyel\_islemi

### **4.1.2. T-SQL PROCEDÜRLERİ**

Genel kullanımı aşağıdaki gibidir:

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
[ { @parameter data_type }
[ VARYING ] [ = default ] [ OUTPUT ] ]
[ ,...n ]
```



**Örnek:** Çalıştırıldığı tarih itibariyle, aldığı kitapları getirmeyen üyelerin ad ve soyadını ekrana yazan bir SP yazalım:

```
CREATE PROCEDURE sp_cezaliUye
AS
DECLARE @buGun DATETIME
SET @buGun = GetDate()
SELECT uyeAdi uyeSoydi FROM odunc
WHERE geldiMi=0 AND VermeTarihi + VermeSuresi < @buGun
```

Daha sonra bu SP

```
EXEC sp_cezaliUye
```

komutu ile çağrılarak çalışması sağlanır ve ekrana aşağıdaki gibi sonuç listesi gelir.

	Adi	soyadi
1	Irem	AKYOL
2	Yusuf	GÖZÜDELI
3	Tugay	ERYATAN

**Şekil 4.2.1.** SP Sonuç Ekranı

### **SP üstünde değişiklik yapmak;**

```
Sp_helptext sp_adi
```

yazılarak SP'nin içeriği görüntülenir. Daha sonra bu içerik T-SQL kod yazma ve geliştirme ekranı olan Query Analyzer kod penceresine yapıştırılıp, CREATE yerine ALTER yazılırsa;

```
ALTER PROCEDURE sp_cezaliUye
AS
DECLARE @buGun DATETIME
SET @buGun = GetDate()
SELECT uyeAdi FROM odunc
WHERE geldiMi=0 AND VermeTarihi + VermeSuresi < @buGun
```

Yazılıp çalıştırılarak SP nin yeni hali kaydedilmiş olur.

**Not :** Daha önce veri tabanı programlarının kod yazmaya gerek kalmadan görsel işlem yapmaya olanak sağlayan programları olduğundan bahsedilmişti. Bir çok işlemi kod yazmaksızın yapmayı sağlayan Enterprice Manager kullanarak sp'ler kolayca değiştirilebilir. Bunun için Enterprice manager ile bir sp'yi seçip çift tıklamak yeterlidir.

### **Bir SP silinmesi;**

```
DROP PROC sp_adi
```

yazılarak SP silinebilir.

**Örnek :** DROP PROC sp\_cezaliUye

### **SP'ye parametre yollama:**

Bazen SP'ler dışarıdan parametre alabilirler:

**Örnek:** Herhangi bir tarih verildiğinde, bu tarihte süresi bittiği halde teslim edilmeyen kitapları bulan bir SP yazalım ancak tarih olarak bu günden daha büyük bir parametre alamasın. Böyle bir durum olduğunda, bu günün tarihini versin.

```
CREATE PROCEDURE sp_cezaliUye_1
@referansTarih DATETIME
AS
DECLARE @buGun DATETIME
SET @buGun = GetDate()
IF @referansTarih > @buGun
SET @referansTarih = @buGun
SELECT uyeAdi FROM odunc
WHERE geldiMi=0 AND VermeTarihi + VermeSuresi < @referansTarih
```

Bazen dışarıdan gelen parametrelerin isteğe bağlı olması istenebilir. Bu durumda DEFAULT değer atama seçeneği kullanılır. Şayet dışarıdan parametreye değer atanmazsa, geçerli değer default atanmış değer olarak alınır ve işlem yapılır:

**Örnek:**

```
CREATE PROCEDURE sp_cezaliUye_2
@referansTarih DATETIME = NULL
-- dışarıdan referans tarihi gelmedi ise, NULL olarak kabul et.
AS
DECLARE @buGun DATETIME
SET @buGun = GetDate()
IF (@referansTarih IS NULL) OR (@referansTarih>@buGun)
SET @referansTarih = @buGun
SELECT uyeAdi FROM odunc
WHERE geldiMi=0 AND VermeTarihi + VermeSuresi < @referansTarih
```

şeklinde yazabiliriz. Daha sonra

```
EXEC sp_cezaliUye_2 @referansTarih='01.01.2003'
```

İle veya

```
EXEC sp_cezaliUye_2
```

Diyerek sp çağırılabilir.

**Not :** SP'de bir parametre için default değer tanımlanırken, bu değer sabit bir değer veya NULL olması gerektiğine dikkat edilmelidir.

**Not :** Bazı durumlarda, SP içerisinde SQL ifadesi bir değişkene atanmak suretiyle durumlara göre dinamik olarak meydana getirip, daha sonra da onu çalıştırabiliriz. Bu durumda da EXEC fonksiyonu kullanılır.

**Örnek :** Kitaplar tablosu üzerinde aşağıdaki parametrelere göre arama yapabilecek bir prosedür yazılmıştır. Parametrelerden gelenler için filtreleme yapılmış ve dışarıdan alınabilecek parametreler: ISBNNo, KitapAdı, Yazarı, Özeti olarak belirlenmiştir.

```
CREATE PROCEDURE sp_kitapBul
@ISBNNo CHAR(16) =NULL,
@KitapAdi VARCHAR(55)=NULL,
@kitapYazarı VARCHAR(55)=NULL,
@KitapOzeti VARCHAR(55)=NULL
AS
DECLARE @sSQL VARCHAR(500)
Set @sSQL= 'SELECT * FROM Kitap WHERE 1=1 '
IF @ISBNNo IS NOT NULL
    SET @sSQL = @sSQL + ' AND ISBNNo = ''' + @ISBNNo + ''''
IF @KitapAdi IS NOT NULL
    SET @sSQL=@sSQL+ ' AND KitapAdi LIKE ''' + @KitapAdi + ''''
IF @KitapYazarı IS NOT NULL
    SET @sSQL=@sSQL+' AND KitapYazarı LIKE ''' + @KitapYazarı+ ''''
IF @KitapOzeti IS NOT NULL
    SET @sSQL = @sSQL+ ' AND KitapOzeti LIKE ''' + @KitapOzeti + ''''
print @sSQL
```

şeklinde yazabiliriz. Daha sonra

EXEC (@sSQL) ile çalıştırılır.

Farklı parametre değerleri ile SP'yi çağırabiliriz:

EXEC Sp\_kitapBul @ISBNNo='12345' veya

EXEC Sp\_kitapBul @ISBNNo='12345', @kitapAdi='Yol' gibi.

### **Sp'den Değer Döndürme:**

OUTPUT opsiyonu başka bir prosedüre değer döndürmeye yarar. Yani bir prosedür kendisini çağıran başka bir prosedüre kendi ürettiği bir çıktı değerini yanıt olarak verebilir.

**Örnek :** 5 adet tamsayı girildiğinde ortalamalarını hesaplayıp, OUTPUT parametresi ile döndürecek bir T-SQL procedür kod parçacığı:

```
CREATE PROC ortalayıcı
@sayi1 smallint,
@sayi2 smallint,
@sayi3 smallint,
@sayi4 smallint,
@sayi5 smallint,
@ortalama smallint OUTPUT
AS
SELECT @ortalama = (@sayi1 + @sayi2 + @sayi3 + @sayi4 + @sayi5) / 5
```

**Bu prosedürün döndürdüğü sonucun alınması:**

Bu prosedürdeki ortalama değerini alabilmek için öncelikle bir değişken tanımlanıp sonra prosedür çalıştırılabilir.

**Örnek :** Output olarak tanımlı parametre bir önceki örnekte en son parametredir. Daha önce tanımlanan değişken buraya verilirse bu değişkende SP'nin ürettiği değer döndürülür. Değeri görmek için daha sonra SELECT komutunu çalıştırılmıştır.

```
DECLARE @sonuc smallint
EXEC ortalayici 1,2,11,20,21,@sonuc OUTPUT
SELECT 'ORTALAMA:',@sonuc
```

Prosedürden değer döndürmek için bir başka seçenek olarak RETURN ifadesi kullanılabilir. Bu şekilde OUTPUT ifadelerini hem prosedür içinde hem de prosedürün çağrıldığı yerde tanımlamak zorunda kalınmadan doğrudan değerler döndürülebilir. Normal olarak SQL Server 0 ile -99 arasında değerleri bu şekilde döndürebilir. Bu değerler, sistem bazlı durumları belirlemek üzere ayrılmış kodlardır. Örneğin sıfır değeri prosedürün çalışmasında bir hata oluşmadığını belirtir. Bu haliyle de elbette kullanıcı tarafından da kullanılabilir. Diğerleri ise muhtelif hata durumlarında kullanıcıyı uyarmak için kullanılırlar veya SQL Server'in ileri sürümlerinde kullanılmak üzere ayrılmışlardır. Kullanıcı, -1 ila -99 haricindeki değerleri döndürebilir.

**Örnek :** Bir kitap ödünç verilmek istendiğinde, bu kitabı öncelikle içeride mi diye kontrol eden bir sp yazılmıştır. Girdi parametre: KitapNo. Daha sonra ödünç verilen kitap no, tarih, üye No , kalma süresi verildiğinde bu kitap dışarıda değil ise yeni bir ödünç verme işlemi başlatan (odunc tablosuna kayıt ekleyen) bir sp yazılmıştır.

Öncelikle, Kitapların durumunu kontrol eden SP'yi oluşturalım: Kitap içeride ise 1, değil ise 0 döndürür.

```
CREATE PROC sp_kitapIcerdeMi
@kitapNo SMALLINT
AS
declare @sonuc INTEGER
IF (SELECT COUNT(*) FROM odunc
WHERE kitapNO=@kitapNo AND geldiMi=0 ) > 0
SET @sonuc=0
else
SET @sonuc=1
return @sonuc
```

yazılır.

Sonra da ödünç bilgisini kaydedecek sp yazılmıştır.

```
CREATE PROC sp_oduncVer
@kitapNo INT,
@vermeTarihi DATETIME ='01.01.2003',
@uyeNo INT,
@vermeSuresi INT = 15
AS
declare @sonuc SMALLINT
EXEC @sonuc=sp_kitapIcerdeMi @kitap
No=@kitapNo
IF (@sonuc=1)
    BEGIN
        INSERT INTO odunc(kitapNo,UyeNo,VermeTarihi,vermeSuresi,geldiMi)
        VALUES(@kitapNo,@uyeNo,@vermeTarihi,@vermeSuresi,0)
    END
ELSE
    BEGIN
        print 'kitap dışarda ödünç verilemedi.....'
    END
```

Test etmek için:

```
EXEC sp_oduncVer @kitapNo=1, @vermeTarihi='01.01.2003',@uyeNo=1
```

Kullanılır ve sonuç alınır.

## 5. SQL PROGRAMLAMADA CURSOR (İMLEÇ) ve TRIGGERS (TETİKLEMELER)

### 5.1 CURSOR (İMLEÇ) VE ÖZELLİKLERİ.

Birden fazla kaydın hafızaya getirilme işlemlerine imleç(cursor) açma denir. İmleç açma, özellikle veritabanındaki tablolardan kayıtların teker teker getirilmesinde faydalı olmaktadır. Kayıtlar teker teker getirilerek üzerinde işlemler yapıp tekrar veritabanına kaydedilebilmektedir. Cursor'lar program içinde Open, Close komutlarıyla açılıp kapatılır ve kayıtlar sırasıyla FETCH komutuyla getirilir.

VTYS'deki imleçler metin editörlerindeki cursorler ile aynı işi yapar; metin editörlerinde o an için cursor nerede ise, oradaki verileri baz alan işlemler yapılabilir. Veritabanı sistemlerinde ise, cursor'ün bulunduğu yerdeki veriler kullanılabilir.

SQL sorgularının sonucunda elde edilen kayıtların tamamına resultset denilir. Bazen bir resultset üstünde gezinilmesi gerekebilir. Bu tür durumlarda, imleçler kullanılır. İmleçler özetle bir select ifadesinin döndürdüğü resultset'e satır satır erişmeye olanak verir.

İmleçlerin kullanılması biraz karmaşık gibi gelebilir ama normalde 5 adımlık ardışık işlemler dizisinden ibarettir.

1. Cursor bir SELECT ifadesi için tanımlanır.
2. Cursor veriler üzerinde gezinilmek üzere, OPEN komutu ile açılır.
3. Resultset'in sonuna gelinceye kadar her seferinde bir kayıt olmak üzere FETCH NEXT komutu ile kayıtlar üstünde ilerlenir.
4. Resultset ile ilgili işlemler sona erdiğinde cursor CLOSE ile kapatılır. Ancak kapatılan cursor henüz hafızada yer kaplamaya devam eder.
5. Cursor ile ilgili işlerimiz bittiği anda hafızadan da silmek için cursor DEALLOCATE ile hafızadan boşaltılır.

#### 5.1.1. PL/SQL CURSORS (İMLEÇLERİ)

Oracle, hafızada imleç işlemleri yapabilmek için yer ayırt eder. İki çeşit imleç vardır:

**1- Kapalı İmleçler(Implicit Cursors) :** Yazılan her SELECT, INSERT, UPDATE ve DELETE komutları için veritabanı tarafından otomatik olarak açılan imleçlerdir. Yazılan her SQL için, SQL'in yazım kontrollerini yapmak ve SQL'i çalıştırmak için hafızadan bir yer ayrılır. Bu ayrılan yer için standart olarak tanımlanan imlece kapalı imleç denir. PL/SQL blokları arasında yazılan SQL komutları için, tanımlanan kapalı imlece ait özellikler geçerli olmaktadır. Yazılan her SQL için tanımlanan kapalı imleçlerin şu özellikleri kullanıma açıktır:

**SQL%ISOPEN** : SQL sonucu eğer imleç açık ise “true” değeri, kapalı ise “false” değeri döndürür.

**SQL%ROWCOUNT** : SQL cümlesi tarafında işlem gören kayıt sayısını görüntüler.

**SQL%FOUND** : SQL sonucu en az bir kayıt işlem görmüşse “true”, hiç kayıt işlem görmemişse “false” döndürür.

**SQL%NOTFOUND** : SQL sonucu eğer hiçbir kayıt işlem görmemişse “true”, en az bir kayıt işlem görmüşse “false” değeri döndürür.

```
DECLARE
    Silinen_kayit_sayisi    number(5);
BEGIN
    delete from isci
    where bolum=10;
    silinen_kayit_sayisi:=SQL%ROWCOUNT;
END;
DECLARE
    delete from isci
    where bolum=10;
    if SQL%FOUND then
        commit;
    else
        rollback;
END;
```

**2- Açık İmleçler (Explicit Cursor)** : Kullanıcı tarafından belirli bir işi yapabilmek için açılan imleçlerdir. Özellikle fazla sayıda kayıtların bulunduğu tablolarda, silme, güncelleme ve benzer işlemlerde çok kullanışlı program parçalarıdır.

```
DECLARE
CURSOR <imleç adı> [(<parametre listesi>)] IS
    SQL cümlesi
Kayıt_tipi_değişkeni    <imleç adı>%ROWTYPE
Değişken_1    NUMBER;
Değişken_2    NUMBER;
BEGIN
    OPEN <imleç adı>
    LOOP
        FETCH <imleç adı> INTO kayıt_tipdeğişkeni;
        EXIT WHEN <imleç adı>%NOTFOUND;
        .....
        komutlar;
    END LOOP;
    CLOSE <imleç adı>;
END;
```



Burada imleç adı PL/SQL blokları içerisinde başka bir değişken adı olarak kullanılıyor olmamalıdır. Parametreler “<parametre adı> <veri tipi>” şeklinde bildirilmelidir. PL/SQL içerisinde kullanılan tüm veri tipleri parametreler için kullanılabilir. (char, varchar2, number, date, boolean ya da number veri tipinin integer, real gibi alt veri tipleri). Aşağıda farklı şekillerde açılmış imleç örnekleri verilmiştir:

**Örnek 1.** CURSOR c1 IS SELECT iscino, isciadi, isi, ucret FROM isci WHERE ucret > 2000;

**Örnek 2.** CURSOR c2 RETURN bolum%ROWTYPE IS

SELECT \* FROM bolum WHERE bolumno = 10;

**Örnek 3.** CURSOR c3 (verilen\_tarih DATE) IS

**Örnek 4.** SELECT iscino, ucret FROM isci WHERE isbatar > verilen\_tarih;

Eğer bir imleç açıldığında kayıtlar üzerinde değişiklik yapılacaksa, imleç tanımından sonra “for update [(<sütun(lar)>)]” şeklinde tanımlama yapılmalıdır. Böyle bir tanımlama yapıldığında imleç içerisindeki kayıtlar kilitlenir ve “commit” komutu uygulanana kadar diğer kullanıcılar bu kayıtlara erişemez.

### 5.1.2. T-SQL CURSORS (İMLEÇLERİ)

Kitaplar listesini ekrana yazdıracak (resultset olarak değil de teker teker) kod parçası:

#### 1.Cursor tanımlanır:

```
DECLARE cursor_adi CURSOR FOR  
SELECT İFADESİ
```

Şeklinde tanımlanır.

```
DECLARE cr_kitapListesi CURSOR FOR  
SELECT kitapNo, kitapAdi  
FROM Kitap  
GO
```

#### 2.Cursor açılır:

```
OPEN cursor_adi  
OPEN cr_kitapListesi  
GO
```

#### 3.Cursor üstünde dolaşmak için FETCH komutundan faydalanılır:

```
FETCH NEXT FROM cursor_adi [INTO icine_doldurulacak_degisken1 [, ....] ]  
DECLARE @kitapNo INTEGER,@kitapAdi  
VARCHAR(55)  
FETCH cr_KitapListesi INTO @kitapNo,  
@kitapAdi  
-- ilk satırın üstüne geldik ve ilk kayıta yer alan iki değeri yazdırdık.  
print @kitapNo  
print @kitapAdi  
go
```

Cursor'ün sona gelip gelmediğini anlamak için @@FETCH\_STATUS ve @@rowcount global fonksiyonlarından faydalanılır.

@@FETCH\_STATUS fonksiyonu, en son çalıştırılan FETCH komutunun sonucu hakkında bize bilgi verir. Bu fonksiyon, şu üç değerden birini verir:

- 0: Bir önceki FETCH komutu başarı ile gerçekleştirildi.
- 1: Bir önceki FETCH komutunda bir hata ile karşılaşıldı.
- 2: Resultsetteki tüm kayıtlar bittiği için en sona gelindi, daha fazla kayıt yer almıyor. (end of resultset)

@@rowcount, bir önceki FETCH komutu icra edildikten sonra resultsette toplam kaç kayıt kaldığını tutar. Hiç FETCH komutu kullanılmadı ise Cursor'ün işaretlediği resultsette toplam kaç kayıt yer aldığını gösterir.

Şimdi artık, kitaplar listesi sonuna kadar yazdırılabilir. Bu iş için WHILE yapısı kullanılır.:

```
DECLARE cr_KitaplarListesi CURSOR FOR
SELECT kitapNo,KitapAdi
DEALLOCATE cursor_adi
DEALLOCATE cr_kitapListesi
```

Projelerin çeşitli aşamalarında, özellikle de dışarıdan veri alınırken, bazı kayıtların bire bir tekrarlandığı olabilir. Bu durumda kayıtlardan sadece birini bırakıp diğerlerini silecek bir Cursor oluşturmak gerekir. Kitap tablosu üzerinde hiçbir indeks alanının olmadığı ve kitapların iki- üç kere aynı bilgilerinin tekrarlandığını varsayalım. (Hatta bazı kitaplar 2, diğerleri 3 diğerleri n kere ...) Bunları silecek bir Cursor aşağıdaki gibidir:

```
DECLARE cr_kitapDurumu CURSOR FOR
SELECT kitap.kitapAdi,odunc.GeldiMi
FROM Kitap
LEFT JOIN odunc ON odunc.KitapNo=Kitap.KitapNo
DECLARE @kitapAdi VARCHAR(25), @Durumu BIT
open cr_kitapDurumu
FETCH NEXT FROM cr_kitapDurumu INTO @kitapAdi, @Durumu
WHILE @@FETCH_STATUS =0
BEGIN
    print @kitapAdi
    IF @Durumu = 0
        print 'dışarıda'
    ELSE
        Print 'içeride'
    FETCH NEXT FROM cr_kitapDurumu INTO @kitapAdi, @Durumu
END
GO
```

Go ile bu kısım çalıştırılır ve daha sonra da Close ile imleç kapatılarak Deallocate ile de hafızadan silinir.

```
CLOSE cr_kitapDurumu
DEALLOCATE cr_kitapDurumu
```

**Örnek :** Kitaplar tablosunda aynı kitap numarasından iki tane olan varsa sildirecek bir cursor yazalım:

```
SET ROWCOUNT 0
DECLARE @kitapNo INTEGER, @tekrarSayisi INTEGER, @sayac BIT
DECLARE crKitaplar CURSOR FOR
SELECT kitapNo, COUNT(kitapNo)-1
    FROM Kitap
    GROUP BY kitapNo
    HAVING COUNT(kitapNo)>1
    ORDER BY kitapNo
--kitapNo bir kereden fazla geçenleri silineceği için doğru kitap numaraları seçilir

OPEN crKitaplar
FETCH NEXT FROM crKitaplar INTO @KitapNo, @tekrarSayisi
WHILE @@FETCH_STATUS=0
BEGIN
    BEGIN
        SET ROWCOUNT @tekrarSayisi
        --tekrar sayısının bir eksiği kadar kayıt silinecek
        DELETE FROM Kitap WHERE kitapNo = @kitapNo
        --sonra tekrardan tüm kayıtlar dikkate alınacak şekilde ROWCOUNT=0 atanıyor
        SET ROWCOUNT 0
    END

    FETCH NEXT FROM crKitaplar INTO @KitapNo, @tekrarSayisi
    --print CAST(@kitapNO VARCHAR(2)) + ' nolu kitap silindi'
END
close crKitaplar
deallocate crKitaplar
```

## 5.2 TRİGGERS (TETİKLEMELER)

Tetiklemeler (Triggers) tıpkı prosedürler gibi veritabanına kaydedilir, fakat program kodu içerisinde çağrılarak işletilmezler. Tetiklemeler bir kere yaratılır ve tanımlanırlar ve sonra veritabanı tarafından otomatik olarak başlatılırlar. Tetiklemeleri başlatan bazı olaylar vardır. Bu olaylar INSERT, UPDATE, DELETE gibi veri işleme dili komutları, veri tanımlama dili komutları, veri tabanı açma-kapama işlemleri, bir kullanıcının bağlanma yada bağlantı kesme işlemleri gibi olaylardır. Bir tetikleme prosedürü üç bölümden oluşur:

- Tetikleme olayı yada komutu
- Tetikleme kısıtlaması
- Tetikleme işlem bölümü

Tetiklemelerin oluşturulmasını göstermek için bir örnek verelim. “isci” tablosundaki herhangi bir işçinin maaşı arttırıldığında maaşı 1000’i geçenleri başka bir tabloya kaydeden bir tetikleme yazalım. Burada işleme maaş arttırıldıktan sonra başlanacaktır. Bu yüzden AFTER UPDATE tetiklemesinin kullanılması gerekir.

```
CREATE OR REPLACE TRIGGER maas_artisi
AFTER UPDATE ON isci
FOR EACH ROW
WHEN (new.ucret > 1000)
BEGIN
INSERT INTO isci_log (isci_no, artis_tarihi, yeni_ucret)
VALUES (:new.iscino, SYSDATE, :new.ucret );
END;
```

Tetiklemelerin bu kısmına AFTER ve BEFORE gibi kelimeler yazılır. AFTER INSERT OR UPDATE gibi iki olayı kapsayan şartlarda yazılabilir. Bu Insert veya Update yaptıktan sonra yapılacak tetikleme anlamına gelir. Buradaki işlemde belirtilen olay; her Insert veya Update yapıldıktan sonra kullanıcı tarafından yeniden yapılmadan veritabanı tarafından otomatik olarak yapılır. Buraya ON DATABASE SHUTDOWN (Veri tabanı kapatıldığında) gibi sistem olayları da yazılabilir.

#### **Tetiklemeleri tasarlarken şu noktalara dikkat etmek gerekir :**

- Tetiklemeleri özel bir operasyon çalıştırıldığında, buna bağlı aktivitelerin çalıştırılmasını garantilemek için kullanılır.
- Bir kullanıcı yada bir veritabanı uygulamasının tetikleme ifadesini kullanmasından bağımsız olmalı, merkezi-global operasyonlar için trigger ifadeleri kullanılmalıdır.
- Daha önceden işleyen fonksiyonlara tekrarlayan triggerler’ı tanımlanır. Örneğin veri bütünlüğü kurallarını sağlayan prosedürler yerine kullanılabilir. Bir tablodan müşteri bilgileri silinirken; müşteriye ait bilgilerin silinen müşteriler tablosuna aktarılması ve müşteriye ait hesap işlemlerinin de kontrol edilerek eğer borç veya alacağı yok ise başka tabloya aktarılarak hesaplar tablosundan silinmesi gibi. Veya bir öğrencinin mezun olması olayını düşünelim. Öğrenci mezun konuma geleceği zaman onun mezun olanlar tablosuna aktarılması ve devam eden öğrenciler tablosundan silinmesini sağlayan BEFORE DELETE trigger’i gibi.

Bir tetiklemenin çalıştırılmadan önce derlenmesi gerektiğinden (dolayısıyla bu tetiklemeye bağlı çalışmalarında derlenmesi gerektiğinden) tetiklemelerin büyüklüğünü sınırlı tutmak gerekir. Tetiklemeleri ortalama 60 satırla sınırlandırmak iyi bir hedeftir. Çünkü küçük boyutlu tetiklemelerin derlenmesinin sistem performansı üzerinde önemli etkileri vardır. Eğer tetiklemenin 60 satırdan fazla tutması gerekiyorsa, bunu bir prosedür içine yazıp, tetikleme içinde ismini kullanarak çağırmak daha iyi bir yoldur. Bu yolla, prosedür bir defa derlenir, böylece tetiklemenin her uyarılışında derlenmesi gerekmez.

## 6. SQL PROGRAMLAMADA TRANSACTION (İŞLEMLER) ve HATA DURUMLARI

### 6.1 TRANSACTION (İŞLEMLER) VE ÖZELLİKLERİ.

Bir banka uygulamasını düşünelim. Bir kullanıcı başka bir kullanıcıya havale yaptığında ne olur? Öncelikle havale yapanın hesap bilgilerinden havale yaptığı miktar düşülür. Ardından alıcının hesabına bu miktar eklenir ve havale gerçekleşmiş olur. Ancak her zaman şartlar istendiği gibi olmayabilir. Örneğin, gönderenin hesabından para düşüldüğü anda elektrik kesilebilir yada program takılabilir. Bu durumda, ne olur? Gönderenin hesabından para düşülmüştür ama Alıcının hesabına da geçmemiştir yani bir kısım paranın sahibinin kimliği kaybedilmiş olur. Bu da sistemin olası durumlar dışında veri kaybetmeye müsait bir hal alması demektir, önlenmesi gerekir.

Daha küçük parçalara ayrılamayan en küçük işlem yığınının Transaction denir. Geçerli kabul edilmesi bir dize işlemlerin tamamının yolunda gitmesine bağlı durumlarda transaction kullanılır. Transaction bloğu ya hep ya hiç mantığı ile çalışır. Ya tüm işlemler düzgün olarak gerçekleşir ve geçerli kabul edilir veya bir kısım işlemler yolunda gitse bile, blok sona ermeden bir işlem bile yolunda gitmese hiçbir işlem olmamış kabul edilir.

Bir Transaction bloğunu işletmenin temel mantığı şu şekildedir:

1. Transaction bloğu başlatılır. Böylece yapılan işlemlerin geçersiz sayılabileceği VTYS 'ye deklare edilmiş olur. Böylelikle VTYS işlemlere başlamadan önceki halin bir kopyasını tempdb 'de tutmaya başlar.
2. Transaction bloğu arasında yapılan her bir işlem bittiği anda başarılı olup olmadığına bakılır, başarılı olmadığı anda geri alım işlemine geçilir (ROLLBACK). Başarılı ise, bir sonraki işleme geçilir.
3. Tüm işlemler tamamlandığı anda COMMIT ile bilgiler yeni hali ile sabitlenir. Başarısız bir sonuç ise ROLLBACK ile en başa alınır ve bilgiler ilk hali ile sabitlenir. Bu örnek için aşağıdaki tablo kullanılacaktır :

Hesap tablosu

#### Alanlar:

HesapNo CHAR(20) NOT NULL PRIMARY KEY  
Adi VARCHAR(55)  
Soyadi VARCHAR(55)  
Sube INTEGER  
Bakiye FLOAT

Genel Yapısı şu şekildedir:

**1.Transaction başlatılır:**

BEGIN TRAN[SACTION] [transaction\_adi]

İle bir transaction başlatılır.



	hesapNo	adi	Soyadı	subeKod	bakiye
1	1	Serap	GÜLSEVEN	102	125000000.0
2	2	Cemile	GÖZÜDEKİ	103	125000000.0
3	3	Musa	ÖZTÜRK	101	100000000.0

**Şekil 6.1.1.** Transaction Uygulanacak Tablo Bilgileri Ekranı

**Örnek:**

```
DECLARE @havaleMiktar FLOAT
DECLARE @aliciHesap VARCHAR(20), @gonderenHesap VARCHAR(20)
SET @aliciHesap='1111122132113'
SET @gonderenHesap='1111122132112'
```

```
SET @havaleMiktar=20000000
-- 20 milyon havale edilecek
```

```
BEGIN TRANSACTION
UPDATE tblHesap
SET bakiye=bakiye - 20000000 WHERE hesapNo=@gonderenHesap

UPDATE tblHesap
SET bakiye=bakiye + 20000000 WHERE hesapNo=@aliciHesap
```

2.İşlem başarılı olursa, COMMIT ile transaction bitirilir. Başarısız olduğunun anlaşılması halinde ROLLBACK komutu ile transaction başarısız olarak bitirilebilir. Yani para havale işlemi hiç yapılmamış gibi en baştaki duruma geri dönülür. Veri tutarlığı için transaction çok önemlidir.

**COMMIT (Sabitleme noktaları: )** Bazen, bir noktaya kadar gelindikten sonra, işlemlerin buraya kadar olanını geçerli kabul etmek isteriz ama, bundan sonraki işlemler için de transaction (geri alabilme seçeneği)ne ihtiyaç duyarız. Bu türden durumlarda sabitleme noktalarından faydalanılır.

Bir sabitleme noktası başlatıldığı anda, en başa dönme seçeneği saklı kalmak üzere, noktanın oluşturulduğu yere de dönme seçeneği sunar

Genel yapısı şu şekildedir:

SAVE TRANSACTION sabitleme\_notkasi\_adi

**Örnek:**

```
BEGIN TRANSACTION
UPDATE tblHesap
SET bakiye = 5000000 WHERE hesapNo='1'

SAVE TRANSACTION svp_kaydet

DELETE FROM tblHesap WHERE HesapNo='1';

ROLLBACK TRAN svp_kaydet;
SELECT * FROM tblHesap;
ROLLBACK TRAN ;
SELECT * FROM tblHesap;
COMMIT
```

**6.2 HATA DURUMLARI**

Hata durumları normalin haricindeki durumlarda nelerin yapılacağını tanımlandığı bölümdür. Belirtilen özel durum oluştuğunda Oracle, PL/SQL komutlarını çalıştırmaya devam etmemekte, onun yerine o özel durumda yapılması tanımlanan işlemleri yapmaya çalışmaktadır. Hata durumları iki türdür: Oracle tarafından önceden tanımlanmış hata durumları ve kullanıcı tanımlı hata durumları.

```
EXCEPTION
WHEN <önceden tanımlanmış hata durumu> THEN
    Komutlar
```

Oracle tarafından tanımlı çok kullanılan hata durumları:

DUP_VAL_ON_INDEX	Tekil olması gereken bir alana bu durumu ihlal eden bir kayıt eklenmeye çalışıldığında ortaya çıkar.
INVLAID_CURSOR	“OPEN” komutu ile açılmamış bir imleç ile işlem yapılmaya çalışıldığında ortaya çıkar.
INVALID_NUMBER	Değişkenin tanımından daha büyük bir sayı değişkene atanmak istendiğinde ortaya çıkar.
NO_DATA_FOUND	SQL sonucu kayıt dönmediği zaman ortaya çıkar.
ZERO_DIVIDE	Sıfıra bölme işleminde ortaya çıkan durumdur.
TOO_MANY_ROWS	Bir kayıt dönmesi gereken SQL ‘den birden fazla kayıt döndüğünde ortaya çıkan durumdur.
VALUE_ERROR	Nümerik veya karakter tipli bir değişkenin diğerinin yerine kullanılmaya çalışıldığı durumdur.
CURSOR_ALREADY_OPEN	Açık bir imlecin tekrar açılmaya çalışıldığı durumdur.
LOGIN_DENIED	Veritabanına yanlış kullanıcı adı ve şifre ile bağlanılmaya çalışıldığı durumdur.
NOT_LOGGED_ON	Veritabanına bağlanmadan SQL cümlesi çalıştırıldığında ortaya çıkan durumdur.
OTHERS	EXCEPTION bölümünde yazılan hata durumlarından hiç biri oluşan hatayla eşleşmediğinde bu durum işleme girer.

Kullanıcıların tanımladığı hata durumları da önceden tanımlı hata durumları ile benzerdir. Farklı olarak kullanıcı bir prosedür çağırıyormuş gibi bu hata durumlarını da çağırmalıdır. Çağırma için “RAISE” komutu kullanılır.

```
DECLARE
.....
BEGIN
.....
  IF ucret<2000 THEN
    RAISE dusuk_maas;
  END IF;
.....
  EXCEPTION
    WHEN dusuk_maas THEN
      Message('Düşük maaşlı işçi');
END;
```



## 7. FORMLAR ve ACCESS PROGRAMINDA FORM OLUŞTURMA

Bu ve bundan sonraki bölümlerde bir veritabanı üzerinde form ve rapor oluşturma konuları anlatılacaktır. Veritabanı üzerinde yapılabilecek işlemler; veritabanına bilgi girilmesi, veri tabanındaki bilginin görüntülenmesi, değiştirilmesi, silinmesi veya listelenmesi olarak adlandırılır. Formlar ve raporlar veri tabanı üzerinde işlem yapmayı sağlarlar.

**Formlar;** veritabanı işlemlerinin ekranda görsel olarak yapılmasını sağlarlar. Formlar olmadan bilgiler ekranda görüntülenemez veya bilgiler üzerinde işlemler yapılamaz. Visual Basic 'de hazırlanan her ekranın bir form olduğunu hatırlayın. 1.sınıf dersi olan entegre ofis dersinde Bölüm-18 de Access programı kapsamında formlardan bahsedilmiş ve nasıl form hazırlandığı anlatılmıştı.

**Raporlar;** veritabanındaki bilgilerden istenilenlerinin ekran veya yazıcıda listelenmesi yani bilgilere ait çıktıların alınmasını sağlarlar. Rapor hazırlanmadan veritabanına ait bilgilerin çıktısı alınamaz. 1.sınıf dersi olan entegre ofis dersinde Bölüm-19 da Access programı kapsamında raporlardan bahsedilmiş ve nasıl rapor hazırlandığı anlatılmıştı.

Veritabanının en genel tanımıyla, kullanım amacına uygun olarak düzenlenmiş veriler topluluğu olduğu daha önce belirtilmişti. Veritabanı bilgileri bünyesinde barındıran bir programdır ama kullanıcıların bu verilere erişmesini, onları kullanmasını ve onlardan yeni veriler elde etmesini sağlayan bir programlama diline sahip değildir.

SQL ve SQL Programlama (PL/SQL ve T-SQL) ile veritabanı üzerinde işlemler yapılabilir ama bu iş için SQL bilmek gerekmektedir. Hazırlanan programları kullanan operatörler için SQL bilmek veya her bilgiye erişmede bir sorgu yapılması ve sorgunun sonucunun beklenilmesi mümkün olmayacağından kullanıcıların sadece bir program yoluyla işlemlerini halletmesi gerekmektedir. Örneğin havale yapmak üzere bankaya giden bir müşterinin havale işlemlerinin SQL ile yapılmaya çalışılmasının mümkün olmaması gibi. Her banka çalışanının SQL bilmesi mümkün değildir ve gerekli de değildir.

Veritabanındaki bilgilerin program haline gelmesi için ayrı programlara ihtiyaç duyulmaktadır. Bunlar Visual Basic, C++, Delphi, PHP, ASP vb. programlama dilleri veya büyük veritabanlarının bu iş için özel hazırlamış olduğu Oracle Forms, Oracle Reports gibi hazır programlardır. Bunlar kullanılarak veritabanı üzerinde işlemler yapılabilir. Kısaca veritabanı işlemleri bir bilgisayar programı halinde kullanıcıya sunulmalıdır ve bunu sağlayan program isimleri yukarıda sayılmıştır.

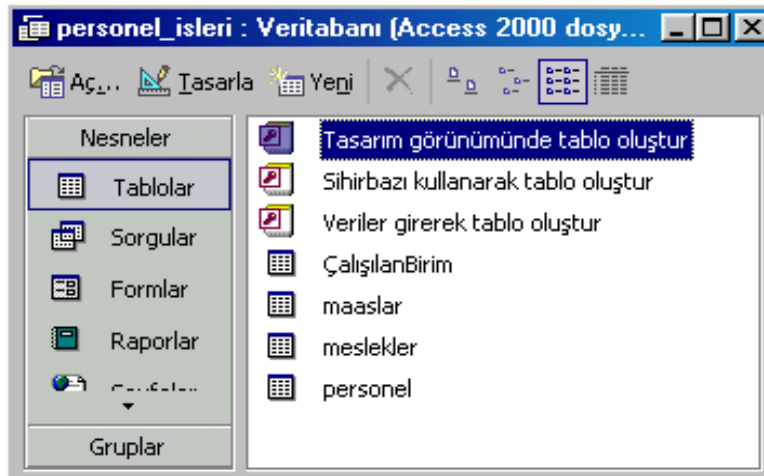
Form ve Rapor oluşturmak için pek çok metot bulunmaktadır. Örneğin bir form Visual Basic programlama dili kullanılarak oluşturulabileceği gibi eğer veritabanı Oracle ise Oracle Forms ile form, Oracle Reports ile de rapor oluşturulabilir.

Büyük veritabanı programlarının form ve rapor hazırlamak için geliştirdiği tools (araç) denen bu programları kullanabilmek için ayrıca onların eğitimlerini de almak gerekmektedir. Bu araçlarla program geliştirmek çok kolaydır. Araç içinde geliştirilmiş sihirbazlar ile çok kısa sürede bir form veya rapor hazırlanabilir. İstenirse sihirbazlar kullanılmadan da kendiniz tasarlayarak form veya raporlar oluşturabilirsiniz.

Bu ders notları kapsamında form ve rapor hazırlama anlatılacaktır. Bu amaçla önce örnek veri tabanı Access programında tasarlanacak ve Access ile form ve rapor hazırlanması, daha sonra Oracle Form ile örnek bir form hazırlanması ve en son aşama da ise Visual Basic ile Access veritabanına erişilerek Form ve rapor hazırlanması anlatılacaktır.

## 7.1 ACCESS İLE FORM OLUŞTURMA

Access programının nasıl kullanıldığını 1.sınıf Entegre Ofis dersinde öğrenmiş olduğumuzdan burada ayrıca Access anlatılmayacaktır. Access 'de tanımlanan tablolar, verilerin yapıları ile tablolar arasındaki ilişki anlatılarak form oluşturulacaktır. Aşağıda Veri tabanı Yönetim Sistemleri I dersinde tasarlamış olduğumuz personel, meslekler, çalışılan birim ve maaşlar tablosundan oluşan örnek personel veritabanının Access programında tanımlanmış hali bulunmaktadır.

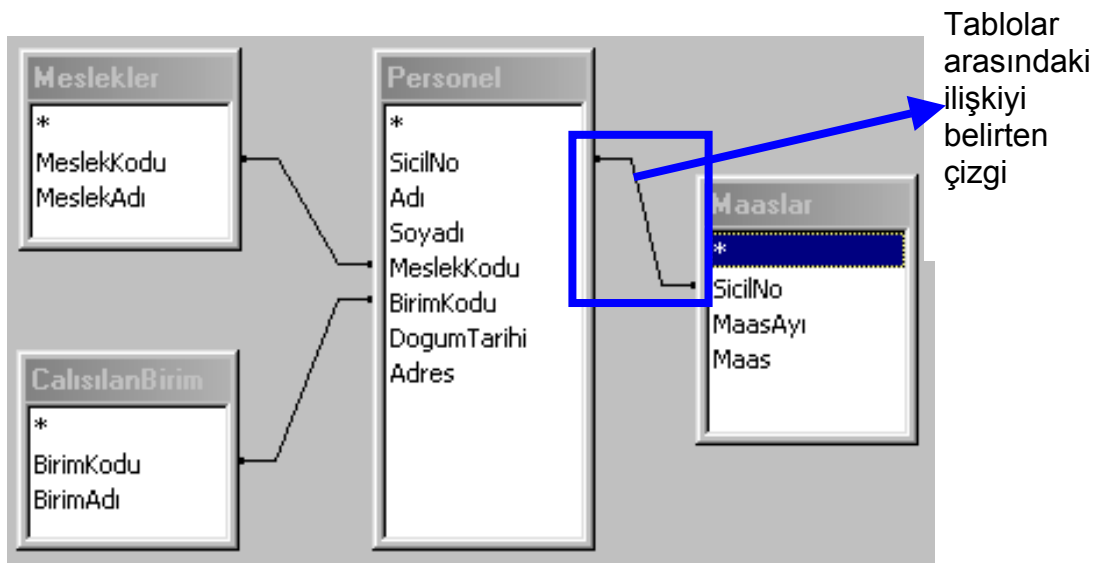


Şekil 7.1.1. Access Veritabanında hazırlanmış Tablolar

Burada 4 adet tablomuz bulunmaktadır. Her tablo ve tablodaki bilgiler aşağıdaki ekranlarda gösterildiği gibidir. Daha sonra bu bilgilerden form ve rapor hazırlanacaktır.

TabloAdı	TabloAlanları	Veri Tipi
Meslekler	MeslekKodu, MeslekAdı	Sayı ( 3) Karakter(20)
ÇalışılanBirim	BirimKodu, BirimAdı	Sayı(3) Karakter(20)
Maaslar	SicilNo, MaasAyı,	Sayı(4) Sayı(2)

<b>Personel</b>	SicilNo, Adı, Soyadı, MeslekKodu, BirimKodu, DoğumTarihi, Adres Maas	Sayı(4) Karakter(15) Karakter(15) Sayı(3) Sayı(3) Tarih Karakter(25) Karakter(12)
-----------------	---	--



Şekil 7.1.2. Örnek Personel Projenin MS Access deki diyagramı

Dört tablo ve tablolar arasındaki ilişkiler yukarıdaki şekilde gösterilmiştir. Mavi çizgi ile tablolar arasındaki ilişkiler gösterilmiştir. Aynı diyagramın bir benzeri SQL-Server veya Oracle veri tabanlarında da hazırlanabilir.

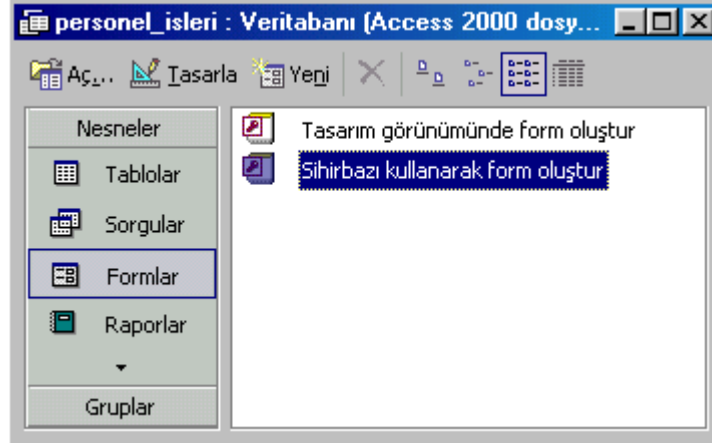
ÇalışılanBirim : Tablo		meslekler : Tablo	
BirimKodu	BirimAdı	MeslekKodu	MeslekAdı
101	Personel	1	Genel Müdür
201	Muhasebe	2	Müdür
301	Bilgi İşlem	3	Müdür Yardımcısı
401	Teknik Servis	4	Şef
*	0	5	Memur
		6	Programcı
		7	Muhasebeci
		8	Tekniker
		9	Teknisyen
		*	0

Şekil 7.1.3. Çalışılan Birim ile Meslekler Tablosu ve Bilgileri

personel : Tablo								maaslar : Tablo			
SicilNo	Adi	Soyadı	Mesle	BirimK	DoğumTarih	Adres		SicilNo	Maas	Maas	
1240	Durmuş	Bozkara	4	101	10.01.1968	Fatih-İstanbul		1240	1	950	
648	Ali	Kantar	2	101	10.01.1968	Fatih-İstanbul		648	1	1250	
117	Fikret	Tatar	5	101	10.01.1968	Taksim-İstanbul		117	1	800	
326	Halil	Kangal	5	101	10.01.1968	Konak-İzmir		326	1	795	
890	Hüseyin	Kantar	5	101	10.01.1968	Konak-İzmir		890	1	815	
1002	Zehra	Yüksek	5	101	10.01.1968	Alsancak-İzmir		1002	1	850	
1290	Kaan	Yalçın	5	101	10.01.1968	Fatih-İstanbul		1290	1	875	
835	Kazım	Akbele	3	101	10.01.1968	Mezitli-Mersin		835	1	1000	
680	Hülya	Bakar	2	201	10.01.1968	Mezitli-Mersin		680	1	1250	
186	Ali	Sümbül	6	201	10.01.1968	Taksim-İstanbul		186	1	1400	
153	Orhan	Duru	7	401	10.01.1968	Çankaya-Ankara		153	1	1100	
245	Fatma	Papatya	7	201	10.01.1968	Çankaya-Ankara		245	1	1050	
345	Serpil	Yücel	5	201	10.01.1968	Alsancak-İzmir		345	1	800	
654	Refik	Erel	5	201	10.01.1968	Taksim-İstanbul		654	1	825	
522	Selim	Arman	7	201	10.01.1968	Beykoz-İstanbul		522	1	1100	
1110	Salih	Bayar	6	301	10.01.1968	Beykoz-İstanbul		1110	1	1400	
178	Fatma	Yalçın	6	301	10.01.1968	Ulus-Anakara		178	1	1450	
898	Mehmet	Gezgin	8	301	10.01.1968	Ulus-Anakara		898	1	875	
1077	Orhan	Sezgin	8	301	10.01.1968	Ulus-Ankara		1077	1	1400	
1121	Nazan	Ulusoy	5	301	10.01.1968	Ulus-Ankara		1121	1	825	
960	Özgen	Öney	2	301	10.01.1968	Çankaya-Ankara		960	1	1200	
590	Özgür	İrmak	3	401	10.01.1968	Taksim-İstanbul		590	1	1000	
772	Nihat	Ünalp	8	401	10.01.1968	Sincan-Anakara		772	1	825	
393	Saliha	Talay	8	401	10.01.1968	Sincan-Anakara		393	1	950	
849	Necia	Alakoç	5	401	10.01.1968	Seyhan-Adana		849	1	775	
967	Fatma	Erdem	5	401	10.01.1968	Merkez-Van		967	1	750	
267	Cengiz	Burma	1	101	10.01.1968	Mezitli-Mersin		267	1	2000	
520	Nuran	Gölbakay	7	201	10.01.1968	Gölbaşı-Ankara		520	1	1150	
456	Galip	Aslan	8	101	10.01.1968	Tece-Mersin		456	1	975	
997	Zehra	Burma	6	301	10.01.1968	Mezitli-Mersin		997	1	1500	
* 0			0	0				* 0	0	0	0
Kayıt: 1 / 30								Kayıt: 1 / 30			

**Şekil 7.1.4. Personel ile Maaşlar Tablosu ve Bilgileri**

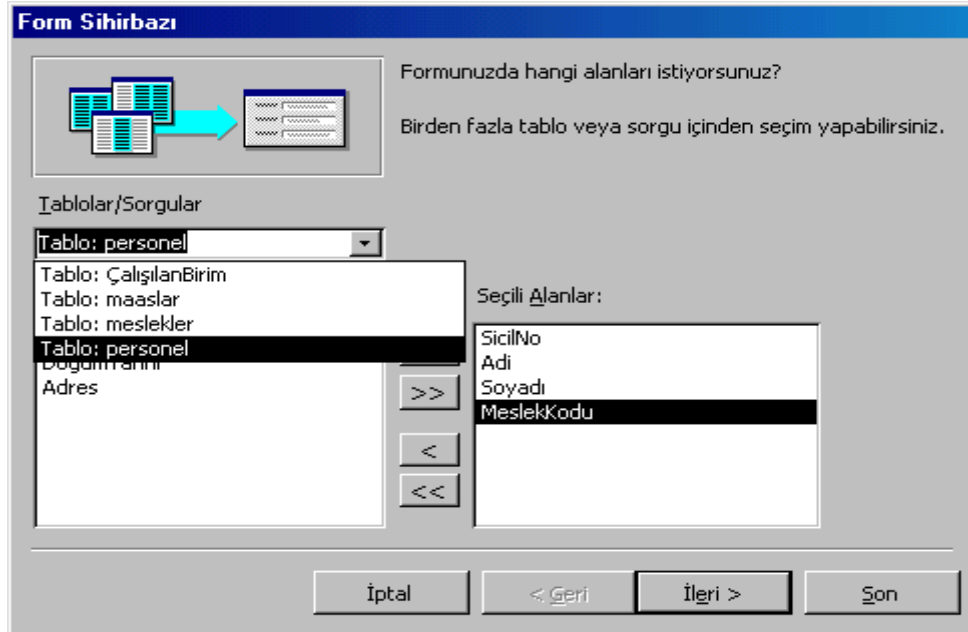
Form ekranın hangi amaçla kullanılacağı daha önce anlatılmıştı. Formlar; tablolara ait bilgi girişlerinin tablolar bölümünde veri sayfası görünümünde yapılabilmesine rağmen formlar hazırlanarak bilgi girişlerinin de yapılabilirdiği ekranlardır. Form ekranı; bize bilgilerin daha görsel bir ekranda girilebilmesini, görüntülenebilmesini, değiştirilebilmesini veya silinebilmesini sağlarlar. Ayrıca kişiye ait özel bilgi giriş ekranlarının oluşturulabilmesini ve Access Veritabanını kullanmayı bilmeyen insanlar tarafından kullanılan ekranın daha görsel ve kolay kullanılmasını sağlayan bir ortam yaratırlar. Formları hazırlayabilmek için Veri tabanı yönetim ekranındaki formlar alanı seçilir.



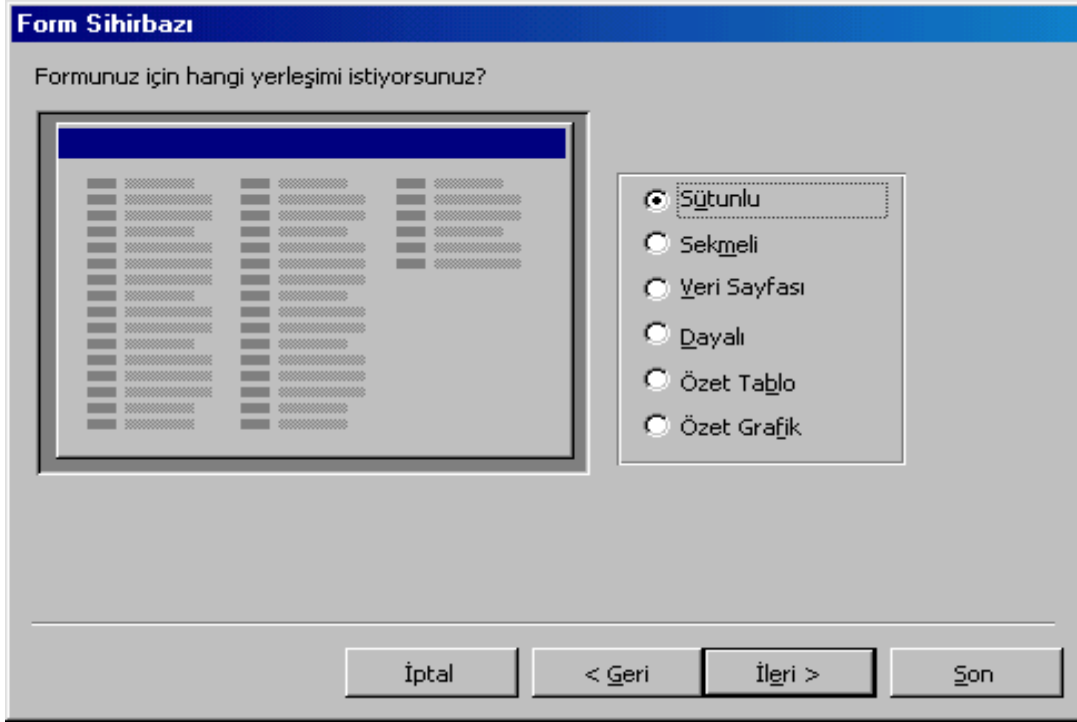
Şekil 7.1.5. Form Sihirbazı ile Form Oluşturma 1. Ekran

Formların anlatıldığı bu bölümde **“Sihirbazı Kullanarak Form Oluştur”** kısmı anlatılmıştır. **“Tasarım görünümünde form oluştur”** profesyonel programcılık bilgisi gerektirdiğinden bu ders notları kapsamında bu seçenek anlatılmamıştır. Form sihirbazı kullanarak form oluştur kısmı sorgu ekranındaki **“Sihirbazı Kullanarak Sorgu Oluştur”** kısmındaki ekrana benzer özellikler taşır.

Ekrana gelen aşağıdaki pencerede form oluşturulacak tablo ve bu tabloya ait alanları önceki yapılan işlemlere benzer şekilde seçili alanlara taşınır ve ileri butonuna tıklanır.



Şekil 7.1.6. Form Sihirbazı ile Form Oluşturma 2. Ekran



**Şekil 7.1.7.** Form Sihirbazı ile Form Oluşturma 3. Ekran

Burada bilgilerin ekrana nasıl yerleştirilmesi istendiği seçilir. Sütunlu, Sekmeli, Veri Sayfası ve Dayalı seçildiğinde her bir ekran için elde edilebilecek görüntülerin yerleşimleri ile ilgili örnek görüntüler **Yerleşimle İlgili Örnekler** başlığı altında ilerleyen sayfalarda verilmiştir. Form sihirbazında form için hangi yerleşim biçimini isteniyorsa buna ait seçenek işaretlenir ve ileri butonuna tıklanılır.

**Not :** Yukarıdaki ekranda bulunan 6 seçeneğin yan kısmındaki yuvarlak seçim butonlarına tıklanıldığı zaman oluşabilecek form şekilleri, size bilgi vermek amacıyla seçim ekranının sol yan tarafında bulunan ekranda görüntülenir. Bu amaçla bu seçimleri deneyiniz ve oluşabilecek form görüntülerini inceleyiniz.

Eğer tabloda bulunan bilgilerin (satırların) her birinin ayrı bir form penceresinde ekrana gelmesi istenirse sütunlu seçilmelidir.

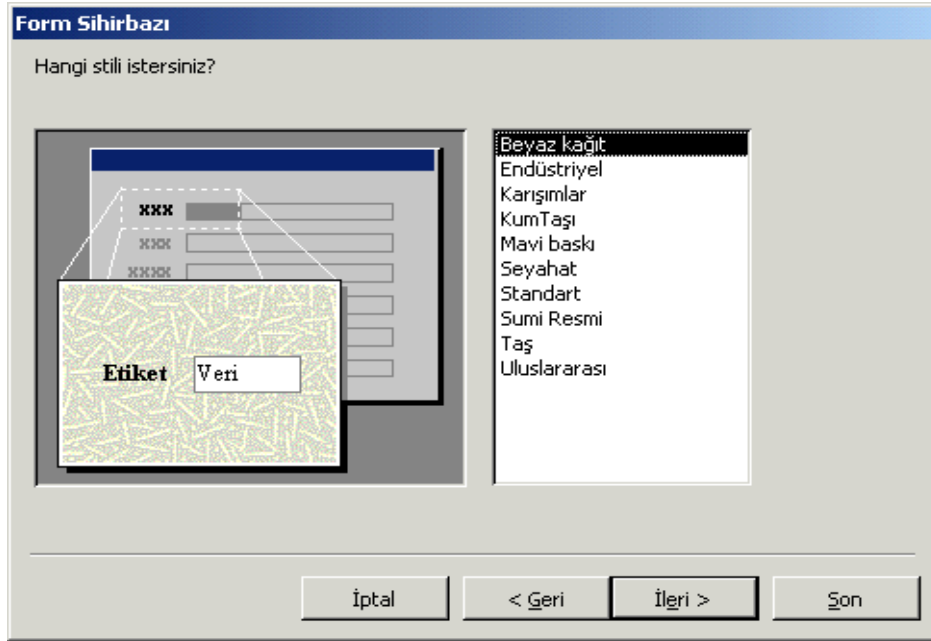
Eğer tabloda bulunan tüm bilgilerin (satırların) hepsinin bir form penceresinde ekrana gelmesi istenirse sekmeli ve veri sayfası seçilmelidir.

Eğer tabloda bulunan bilgilerin (satırların) her birinin form penceresi üzerinde ekrana yan yana olarak gelmesi istenirse dayalı seçilmelidir.

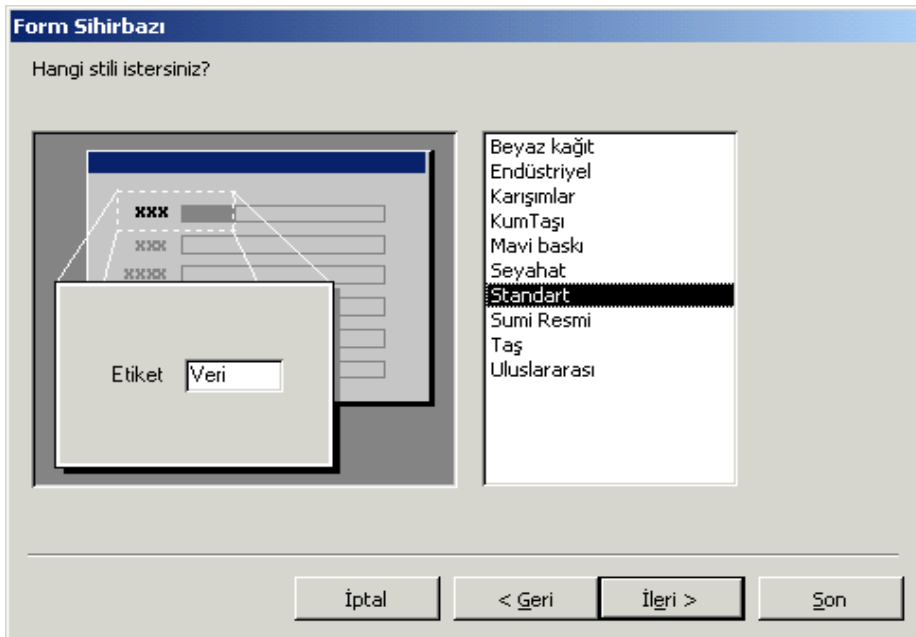
Eğer tabloda bulunan bilgilerin (satırların) her birinin sütunlarına göre form penceresi üzerinde toplamsal bilgilerinin tablosal olarak ekrana gelmesi istenirse Özet Tablo seçilmelidir.

Eğer tabloda bulunan bazı alanların (satırların) her birinin rakamsal form penceresi üzerinde grafiksel olarak ekrana gelmesi istenirse Özet Grafik seçilmelidir.

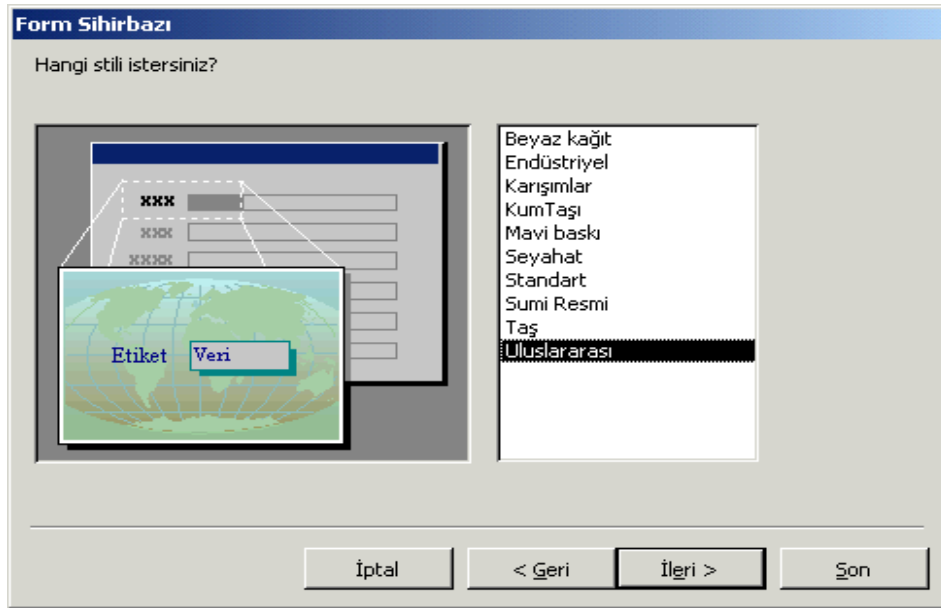
İleri butonuna tıklandıktan sonra formun hangi stil ile ekrana gelmesi gerektiği sorusuna cevap verilir. Burada access içerisinde bulunan hazır form stilleri ekrana gelir ve bu stillerden istenileni mouse ile üzerine tıklanılarak seçilir. Seçilen stil ekrana gelen pencerenin sol yan tarafında görüntülenir. Aşağıda örnek stil ekranları görülmektedir.



Şekil 7.1.8. Form Sihirbazı ile Form Oluşturma 4. Ekran Stil Örneği 1 (Beyaz Kağıt)

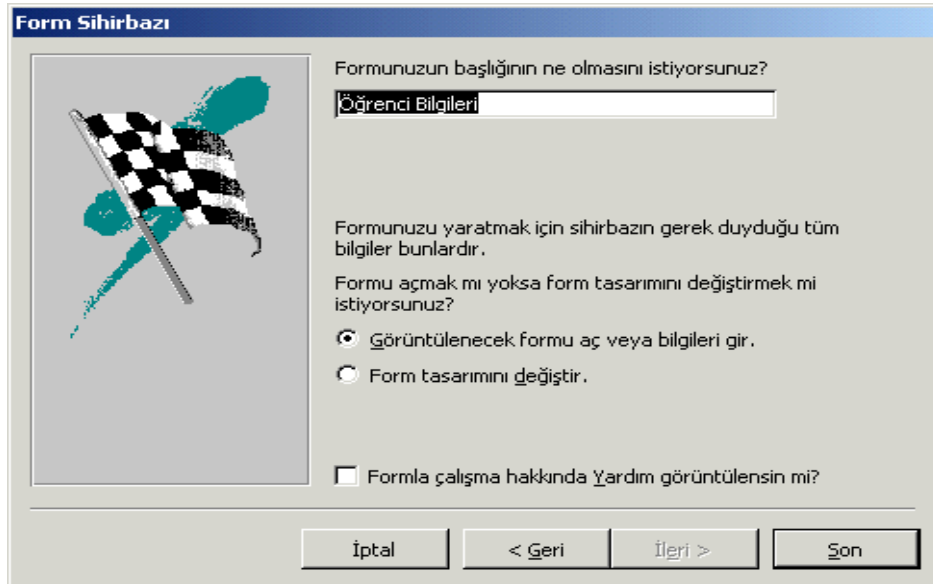


Şekil 7.1.9. Form Sihirbazı ile Form Oluşturma 5. Ekran Stil Örneği 1 (Standart)



Şekil 7.1.10. Form Sihirbazı ile Form Oluşturma 6. Ekran Stil Örneği 1 (Uluslararası)

Burada hangi stili kullanmak isteniyorsa o stil seçildikten sonra her sihirbaz ekranında olduğu gibi bir sonraki aşamaya geçmek için ileri butonuna tıklanır.



Şekil 7.1.11. Form Sihirbazı ile Form Oluşturma 7. Ekran

Hazırlanan formun hangi isimle kaydedileceği bu ekranda belirlenir. Access tarafından otomatik olarak bir form ismi ekrana gelir. Bu ismi sorgu ekranında olduğu gibi aynen kabul edebilir veya değiştirebilirsiniz. Daha Sonra Son butonuna tıklanılarak sihirbaz sona erdirildiğinde ekrana hazırlamış olan form ekranı gelir.

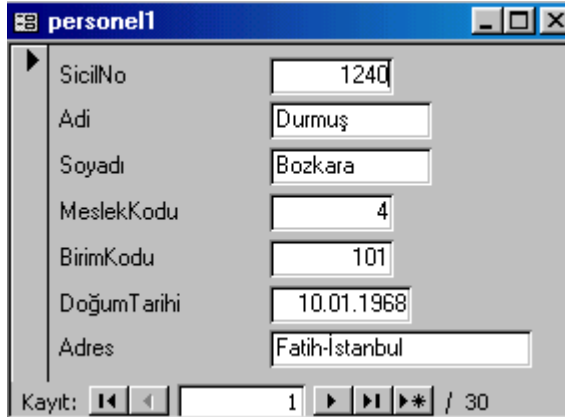




Şekil 7.1.12. Oluşturulan Form Ekranı.

### Yerleşimle İlgili Örnekler :

#### Sütunlu :



Şekil 7.1.12. Sütunlu Yerleştirilmiş Form Örneği Ekranı.

#### Sekmeli:



SicilNo	Adi	Soyadı	MeslekKodu	BirimKodu	DoğumTarihi	Adres
1240	Durmuş	Bozkara	4	101	10.01.1968	Fatih-İstanbul
648	Ali	Kantar	2	101	10.01.1968	Fatih-İstanbul
117	Fikret	Tatar	5	101	10.01.1968	Taksim-İstanbul
326	Halil	Kangal	5	101	10.01.1968	Konak-İzmir
890	Hüseyin	Kantar	5	101	10.01.1968	Konak-İzmir
1002	Zehra	Yüksek	5	101	10.01.1968	Alsancak-İzmir
1290	Kaan	Yalçın	5	101	10.01.1968	Fatih-İstanbul

Şekil 7.1.13. Sekmeli Yerleştirilmiş Form Örneği Ekranı.

### Veri Sayfası

SicilNo	Adi	Soyadı	MeslekKod	BirimKodu	DoğumTarihi	Adres
1240	Durmuş	Bozkara	4	101	10.01.1968	Fatih-İstanbul
648	Ali	Kantar	2	101	10.01.1968	Fatih-İstanbul
117	Fikret	Tatar	5	101	10.01.1968	Taksim-İstanbul
326	Halil	Kangal	5	101	10.01.1968	Konak-İzmir
890	Hüseyin	Kantar	5	101	10.01.1968	Konak-İzmir
1002	Zehra	Yüksek	5	101	10.01.1968	Alsancak-İzmir
1290	Kaan	Yalçın	5	101	10.01.1968	Fatih-İstanbul
835	Kazım	Akbele	3	101	10.01.1968	Mezitli-Mersin
680	Hülya	Bakar	2	201	10.01.1968	Mezitli-Mersin
186	Ali	Sümbül	6	201	10.01.1968	Taksim-İstanbul
153	Orhan	Duru	7	401	10.01.1968	Çankaya-Ankara
245	Fatma	Papatya	7	201	10.01.1968	Çankaya-Ankara
345	Serpil	Yücel	5	201	10.01.1968	Alsancak-İzmir
654	Refik	Erel	5	201	10.01.1968	Taksim-İstanbul
522	Selim	Arman	7	201	10.01.1968	Beykoz-İstanbul
1110	Salih	Bayar	6	301	10.01.1968	Beykoz-İstanbul
178	Fatma	Yalçın	6	301	10.01.1968	Ulus-Ankara
898	Mehmet	Gezgin	8	301	10.01.1968	Ulus-Ankara

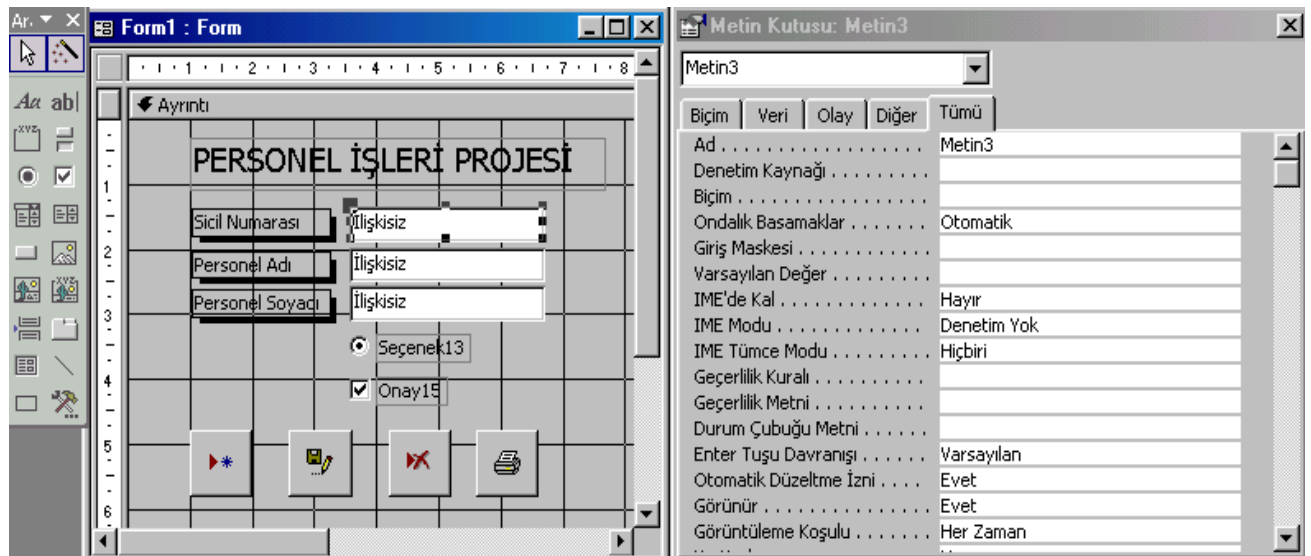
Şekil 7.1.14 . Veri Sayfası Yerleştirilmiş Form Örneği Ekranı.

### Dayalı


SicilNo	Adi	Soyadı	MeslekKodu	BirimKodu
1240	Durmuş	Bozkara	4	101
DoğumTarihi				
10.01.1968				
Adres				
Fatih-İstanbul				

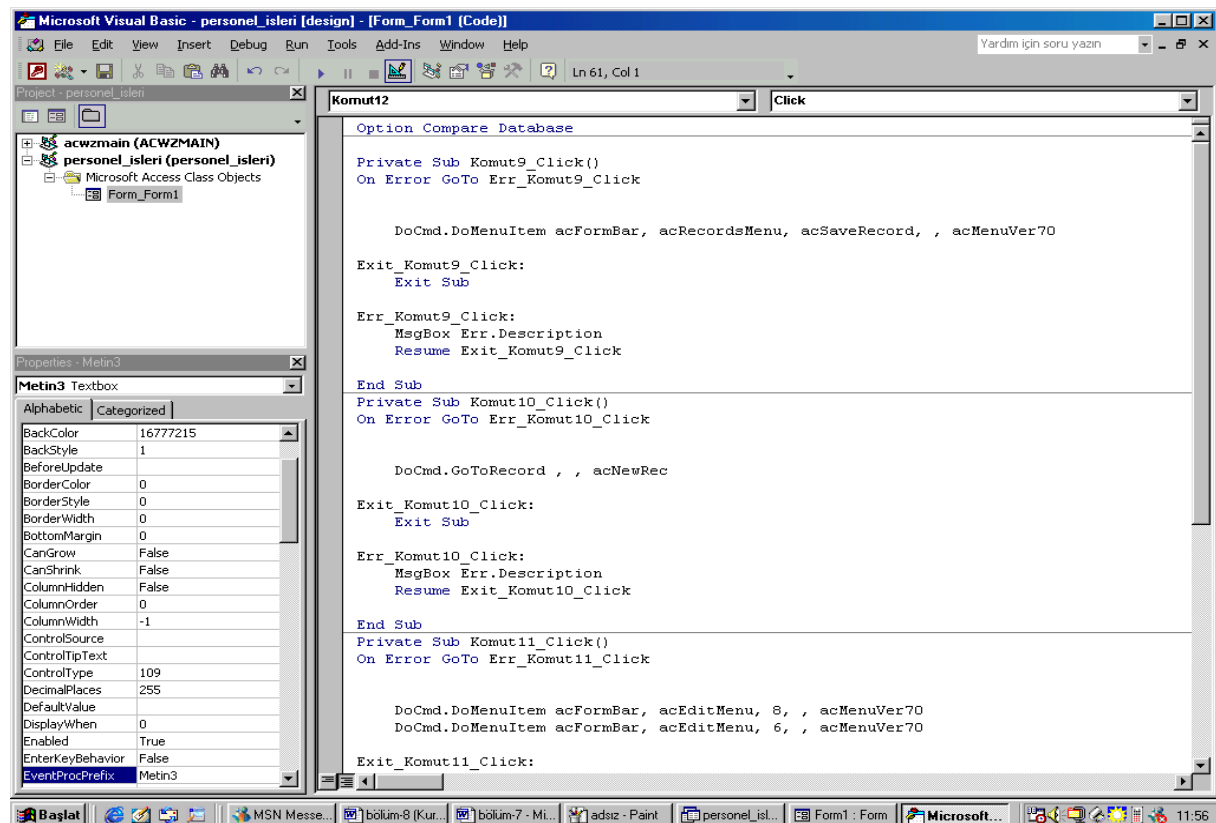
Şekil 7.1.15. Dayalı Yerleştirilmiş Form Örneği Ekranı.

**“Tasarım görünümünde form oluştur”** profesyonel programcılık bilgisi gerektirdiğinden bu ders notları kapsamında bu seçeneğin anlatılmayacağı daha önce söylenmişti. Bir ön bilginiz olması açısından; aşağıda tasarım görünümünde form oluşturma ekranı ve ekran üzerinde tanımlanmış araç kutusu elemanları bulunmaktadır. Bu ekran Visual Basic, Delphi vb. görsel programlama yapan programlama dillerindeki form tasarımından pek farklı değildir. Burada form tasarımı yapılacak ekran, formda kullanılacak textbox, Label, command button gibi elemanların bulunduğu araç kutusu ve elemanların özelliklerinin ayarlandığı ekran bulunmaktadır.



### Şekil 7.1.16. Tasarım Görünümünde Form Oluşturma Ekranı

Access veritabanında tasarım ile form oluştururken gerektiğinde kod da yazılabilir. Access programı kod yazmak için Visual Basic kullanır. Kod yazım ekranına geçmek için tasarım ekranının üst kısmında bulunan  kod araç çubuğuna basılır ve gereken kodlar yazılabilir.



**Şekil 7.1.17.** Tasarım Görünümünde Form Oluşturma Kod Yazım Ekranı

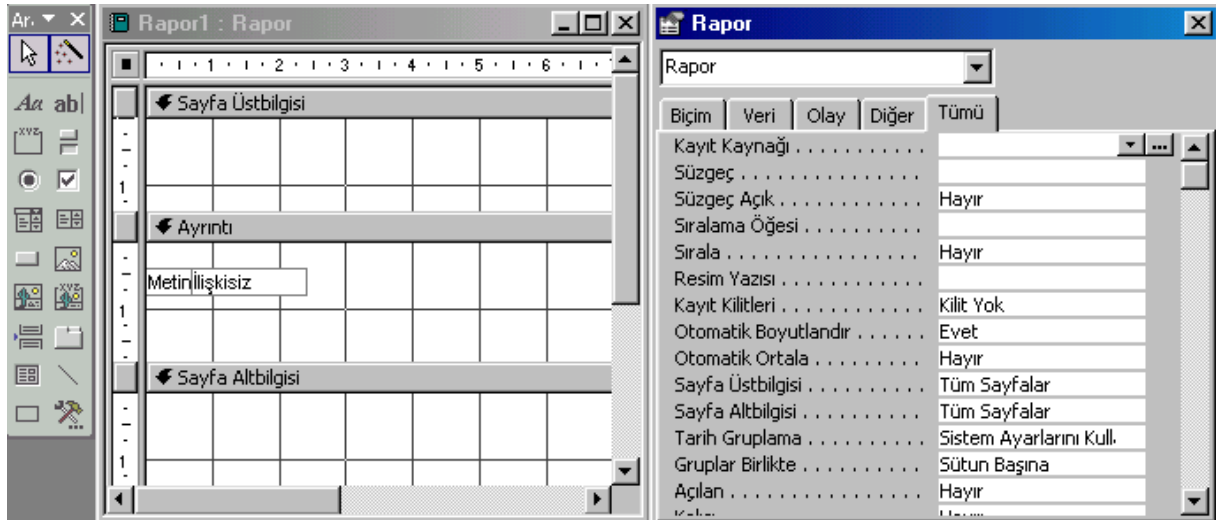
## 8. RAPORLAR VE ACCESS PROGRAMINDA RAPOR OLUŞTURMA

### 8.1 ACCESS İLE RAPOR OLUŞTURMA

Raporlar; tablolardaki ve hazırlanan sorgulardaki bilgilerin istenilen düzenlemelere göre ekran veya yazıcıdan liste halinde alınabilmesi sağlayan bir ortamdır. Raporları hazırlayabilmek için Veri tabanı yönetim ekranındaki raporlar alanı seçilir.

Raporlar kısmında **“sihirbazı kullanarak rapor oluştur”** kısmını seçilerek rapor oluşturma kısmı anlatılmıştır. **“Tasarım görünümünde rapor oluştur”** profesyonel programcılık bilgisi gerektirdiğinden bu ders notları kapsamında anlatılmamıştır.

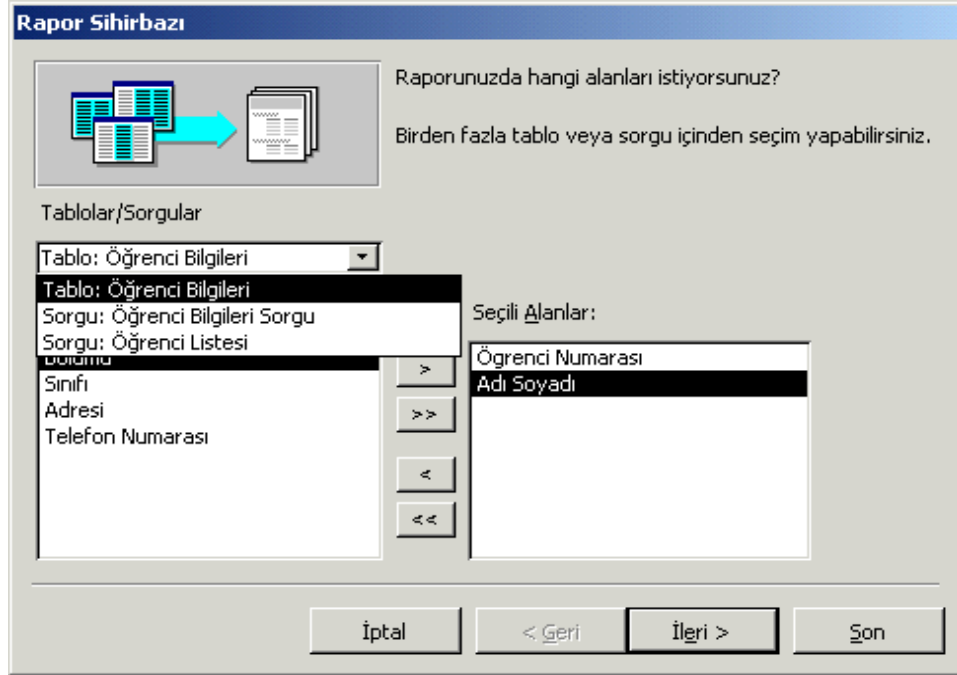
Bir ön bilginiz olması açısından; aşağıda tasarım görünümünde rapor oluşturma ekranı ve ekran üzerinde tanımlanmış araç kutusu elemanları bulunmaktadır. Bu ekran Visual Basic, Delphi vb. görsel programlama yapan programlama dillerindeki rapor tasarımından pek farklı değildir. Aslında Access deki form hazırlama ekranına da benzer özellikler taşır. Burada rapor tasarımı yapılacak ekran, raporda kullanılacak textbox, Label, command button gibi elemanların bulunduğu araç kutusu ve elemanların özelliklerinin ayarlandığı ekran bulunmaktadır.



Şekil 8.1.1. Tasarım Görünümünde Rapor Oluşturma Ekranı

Rapor sihirbazı kullanarak rapor oluştur kısmı **“Sihirbazı Kullanarak Sorgu Oluştur”** ve **“Sihirbazı Kullanarak Form Oluştur”** kısmındaki ekranlara benzer özellikler taşır.

Aşağıda ekrana gelen pencerede rapor oluşturulacak tablo ve bu tabloya ait alanlar sorgu ve form oluşturmaya benzer şekilde seçili alana taşınır ve ileri butonuna tıklanılarak rapor oluşturulmaya başlanır.



**Rapor Sihirbazı**

Raporunuzda hangi alanları istiyorsunuz?  
Birden fazla tablo veya sorgu içinden seçim yapabilirsiniz.

Tablolar/Sorgular

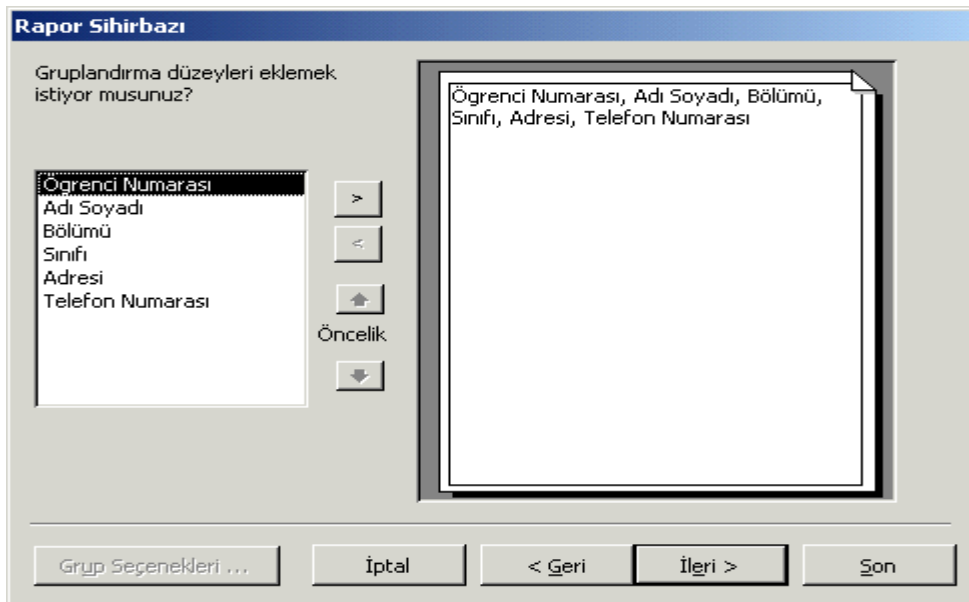
Tablo: Öğrenci Bilgileri  
Sorgu: Öğrenci Bilgileri Sorgu  
Sorgu: Öğrenci Listesi  
Bölümü  
Sınıfı  
Adresi  
Telefon Numarası

Seçili Alanlar:

Öğrenci Numarası  
Adı Soyadı

İptal < Geri İleri > Son

Şekil 8.1.2. Rapor Sihirbazı İle Rapor Oluşturma 1. Ekran



**Rapor Sihirbazı**

Gruplandırma düzeyleri eklemek istiyor musunuz?

Öğrenci Numarası  
Adı Soyadı  
Bölümü  
Sınıfı  
Adresi  
Telefon Numarası

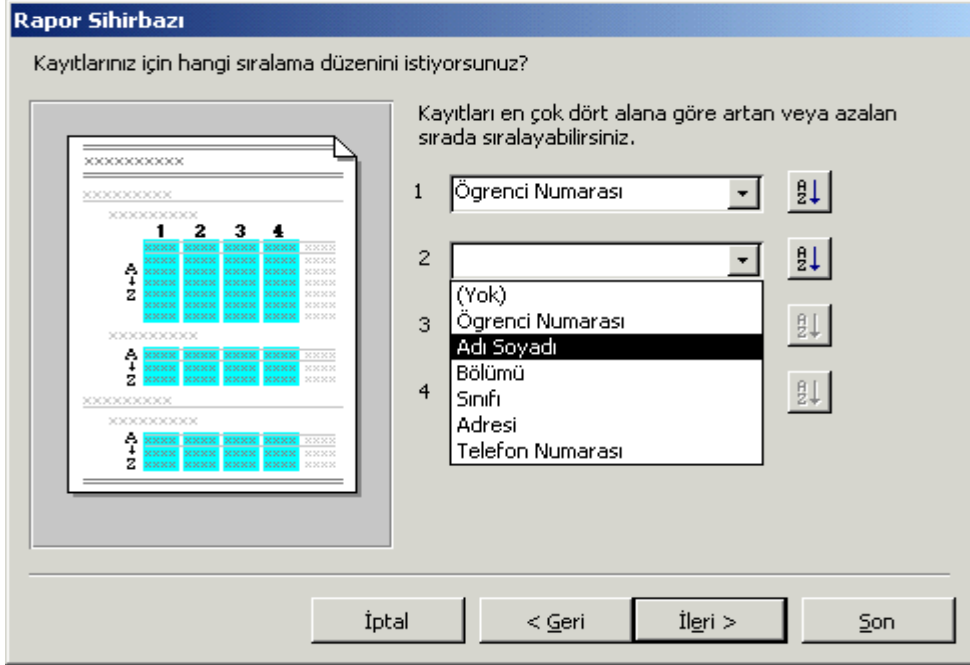
Öncelik

Öğrenci Numarası, Adı Soyadı, Bölümü, Sınıfı, Adresi, Telefon Numarası

Grup Seçenekleri ... İptal < Geri İleri > Son

Şekil 8.1.3. Rapor Sihirbazı İle Rapor Oluşturma 2. Ekran

Bir sonraki adım için İleri butonuna tıklanır.

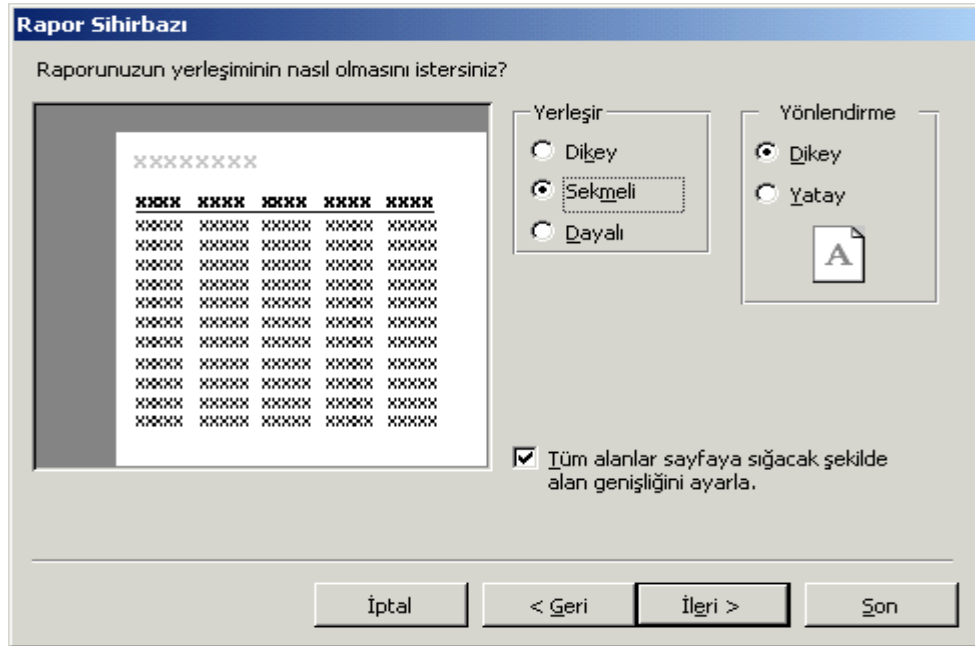


Şekil 8.1.4. Rapor Sihirbazı İle Rapor Oluşturma 3. Ekran

Bu alanda istenilen bir bilgiye göre bir sıralama yapılıp yapılmayacağı sorulmaktadır. Örneğin 4. alan olan sınıflara göre bir sıralama yapılabilir. Sıralamalar birden fazla alan içinde yapılabilir. Örneğin önce bölümlere, sonra da bölümün sınıflarına göre bir sıralama yapılırsa önce bölümler sonra da her bölüm kendi içerisinde sınıflarına göre sıralanır.

Sıralama birden fazla alan için yapılırsa önce 1. alan sıralanır sonra 2. alan 1. alana göre sıralanır. Örneğin 1. sıralama alanı bölüm 2. sıralama alanı sınıf olarak seçilirse öğrenciler bölümlerine göre sınıf sınıf sıralı olarak listelenir. Yukarıdaki ekranda da görüldüğü gibi bilgiler öğrencinin numarasına göre sıralanmıştır. İstenirse Ad-Soyad'a göre de sıralama yapılabilir. Yukarıdaki ekranın 1. nolu alanında seçilen bilginin yan tarafında bulunan "**A-Z**" butonu bize bilginin artan veya azalan sırada gelmesini sağlamaktadır. Buradaki örnekte numara alanı A-Z seçildiğinde öğrenciler numaralarına göre en küçük numaradan en büyük numaraya göre bir sıralanırlar. Eğer bu butona bir kez daha tıklanırsa bu alanın "**Z-A**" olarak değiştiği görülür. Bu durumda numaralar en büyük rakamlı numaralı öğrenciden en küçük rakamlı numaralı öğrencilere göre sıralanır.

Eğer seçilen alan ad-soyad'a göre bir sıralama içeriyorsa ve "**A-Z**" seçilmiş ise bu sefer öğrencilerin ad-soyadı A harfinden Z harfine doğru sıralanır. A-Z butonuna mouse ile bir kez tıklandığında bu sefer tam tersi olan Z harfinden A harfine göre azalan bir sıralama yapılır. Yani sıralanacak olan alan harflerden oluşuyorsa bilgiler A-Z, rakamlardan oluşuyorsa bilgiler rakamsal büyüklüklerine göre A-Z seçeneğinden hangisi seçili ise ona göre sıralanırlar.



Raporunuzun yerleşiminin nasıl olmasını istersiniz?

Yerleşir

- ☐ Dikey
- ☒ Sekmeli
- ☐ Dayalı

Yönlendirme

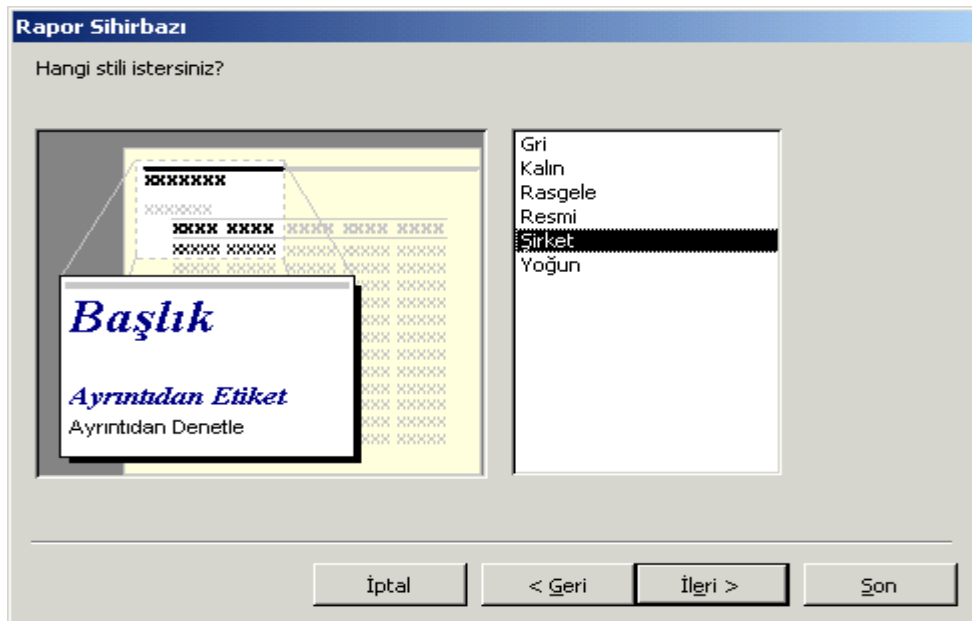
- ☒ Dikey
- ☐ Yatay

☒ Tüm alanlar sayfaya sığacak şekilde alan genişliğini ayarla.

İptal < Geri İleri > Son

Şekil 8.1.5. Rapor Sihirbazı İle Rapor Oluşturma 4. Ekran

Bu ekranda raporun yerleşimi ile ilgili bilgilerin seçimi yapılır. Yerleştir de tıpkı form ekranında olduğu gibi bilgilerin alt alta mı? Yoksa yan yana mı? Olacağı seçilir. Yönlendir bölümünde ise raporun yazıcı kağıdına yerleşme durumunun yatay mı? Yoksa dikey mi? Olacağına karar verilir.



Hangi stili istersiniz?

Gri  
Kalin  
Rasgele  
Resmi  
**Sirket**  
Yoğun

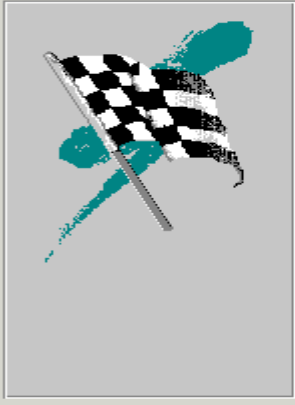
Başlık  
Ayrıntıdan Etiket  
Ayrıntıdan Denetle

İptal < Geri İleri > Son

Şekil 8.1.6. Rapor Sihirbazı İle Rapor Oluşturma 5. Ekran

Rapor stili seçilir.

**Rapor Sihirbazı**



Raporunuzun başlığının ne olmasını istiyorsunuz?

Raporunuzu yaratmak için sihirbazın gerek duyduğu tüm bilgiler bunlardır.

Raporu önizlemek mi yoksa rapor tasarımını değiştirmek mi istiyorsunuz?

☒ Raporu önizle.

☐ Rapor tasarımını değiştir.

☐ Raporla çalışma konusunda Yardım görüntülensin mi?

Şekil 8.1.7. Rapor Sihirbazı İle Rapor Oluşturma 6. Ekran

Raporun hangi isimle kaydedileceği yazılır. Sorgu ve form ekranlarında olduğu gibi burada da otomatik gelen isim kabul edebileceği gibi rapora yeni bir isim verilerek de kaydedilebilir.

*Öğrenci Bilgileri*

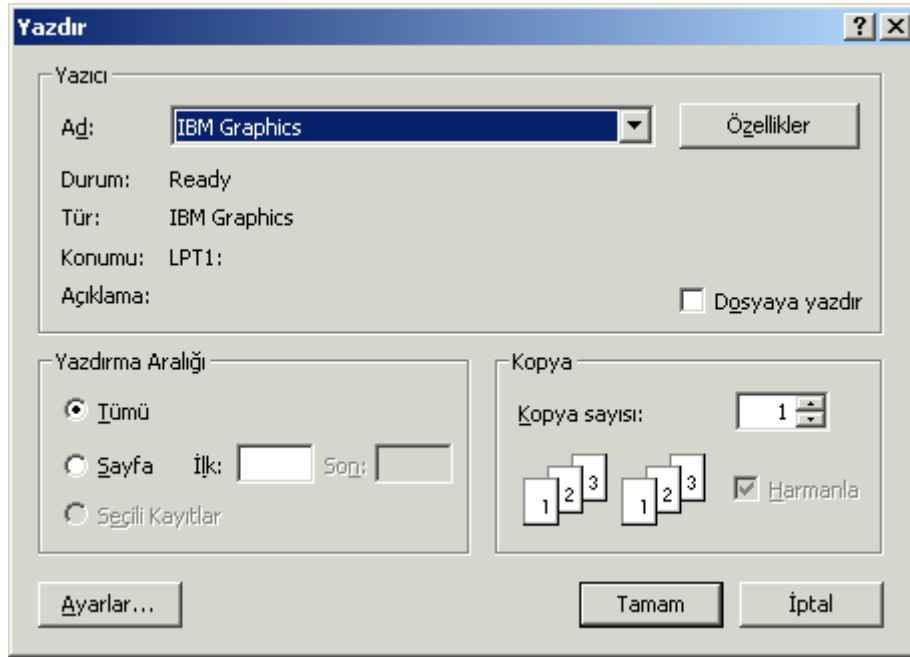
Sıra No	Öğrenci Adı	Soyadı	Doğum Yılı	Sınıf	Okul	Öğretmen Adı
1	Öğrenci	Adı	1990	1	Okul	Öğretmen
2	Öğrenci	Soyadı	1991	2	Okul	Öğretmen
3	Öğrenci	Adı	1992	3	Okul	Öğretmen
4	Öğrenci	Soyadı	1993	4	Okul	Öğretmen
5	Öğrenci	Adı	1994	5	Okul	Öğretmen
6	Öğrenci	Soyadı	1995	6	Okul	Öğretmen
7	Öğrenci	Adı	1996	7	Okul	Öğretmen
8	Öğrenci	Soyadı	1997	8	Okul	Öğretmen
9	Öğrenci	Adı	1998	9	Okul	Öğretmen
10	Öğrenci	Soyadı	1999	10	Okul	Öğretmen
11	Öğrenci	Adı	2000	11	Okul	Öğretmen
12	Öğrenci	Soyadı	2001	12	Okul	Öğretmen
13	Öğrenci	Adı	2002	13	Okul	Öğretmen
14	Öğrenci	Soyadı	2003	14	Okul	Öğretmen
15	Öğrenci	Adı	2004	15	Okul	Öğretmen
16	Öğrenci	Soyadı	2005	16	Okul	Öğretmen
17	Öğrenci	Adı	2006	17	Okul	Öğretmen
18	Öğrenci	Soyadı	2007	18	Okul	Öğretmen
19	Öğrenci	Adı	2008	19	Okul	Öğretmen
20	Öğrenci	Soyadı	2009	20	Okul	Öğretmen

3 / 3 Sayfa 2008

Şekil 8.1.8. Rapor Sihirbazı ile Oluşturulan Raporu Önizleme Ekranı.



Rapor baskı önizleme den sonra bir sorun yoksa yazdır seçeneği ile yazıcıdan alınabilir. Yazdır seçeneğinde yapılacaklar diğer office programlarında olduğu gibidir.



**Şekil 8.1.9.** Raporu Yazdır Ekranı

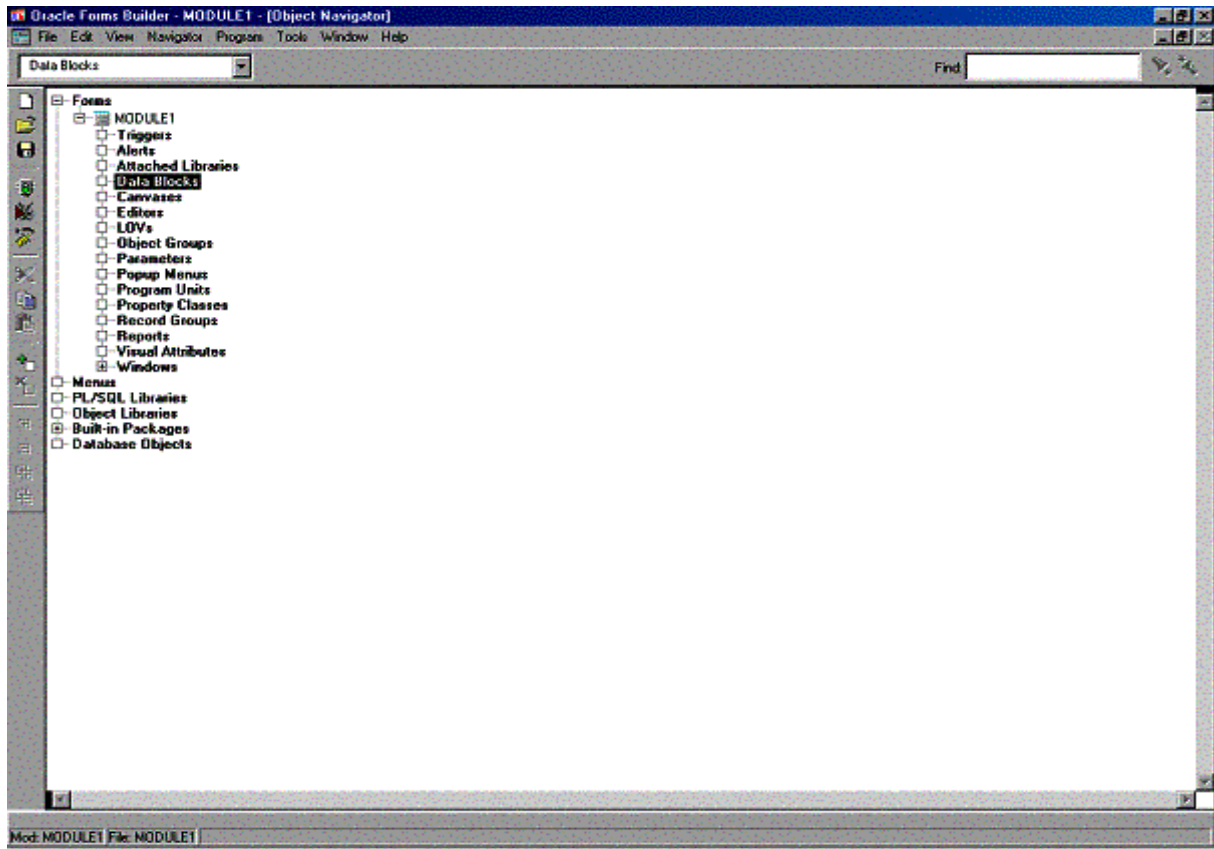
## 9. FORMLAR ve ORACLE FORMS PROGRAMINDA FORM OLUŞTURMA

Bu bölümde Oracle Forms programı ile örnek bir form hazırlanması anlatılacaktır.

### 9.1 ORACLE FORMS ile FORM Oluşturma

Nasıl Delphi programının programlama dili Pascal, C++ Builder'ın ki C++ ise Forms Builder'ın programlama dili ise PL-SQL'dir. Kapsamlı bir form yapılmak istenirse; PL-SQL dilini iyi bilmek gerekmektedir. Aslında Oracle Forms programı da kapsamlı bir program olup kullanılmasını öğrenmek gerekmektedir. Bu amaçla Oracle Firması çeşitli eğitimler düzenlemektedir. SQL 'in ise PL/SQL için zaten bilinmesi gerekmektedir. Oracle veritabanındaki kullanıcılardan scott/tiger'in EMP tablosu kullanılarak basit bir form hazırlayalım.

Forms Builder ilk açıldığında ekrana aşağıdaki gibi bir görüntü gelir ve burada sırasıyla şu adımlar uygulanır.

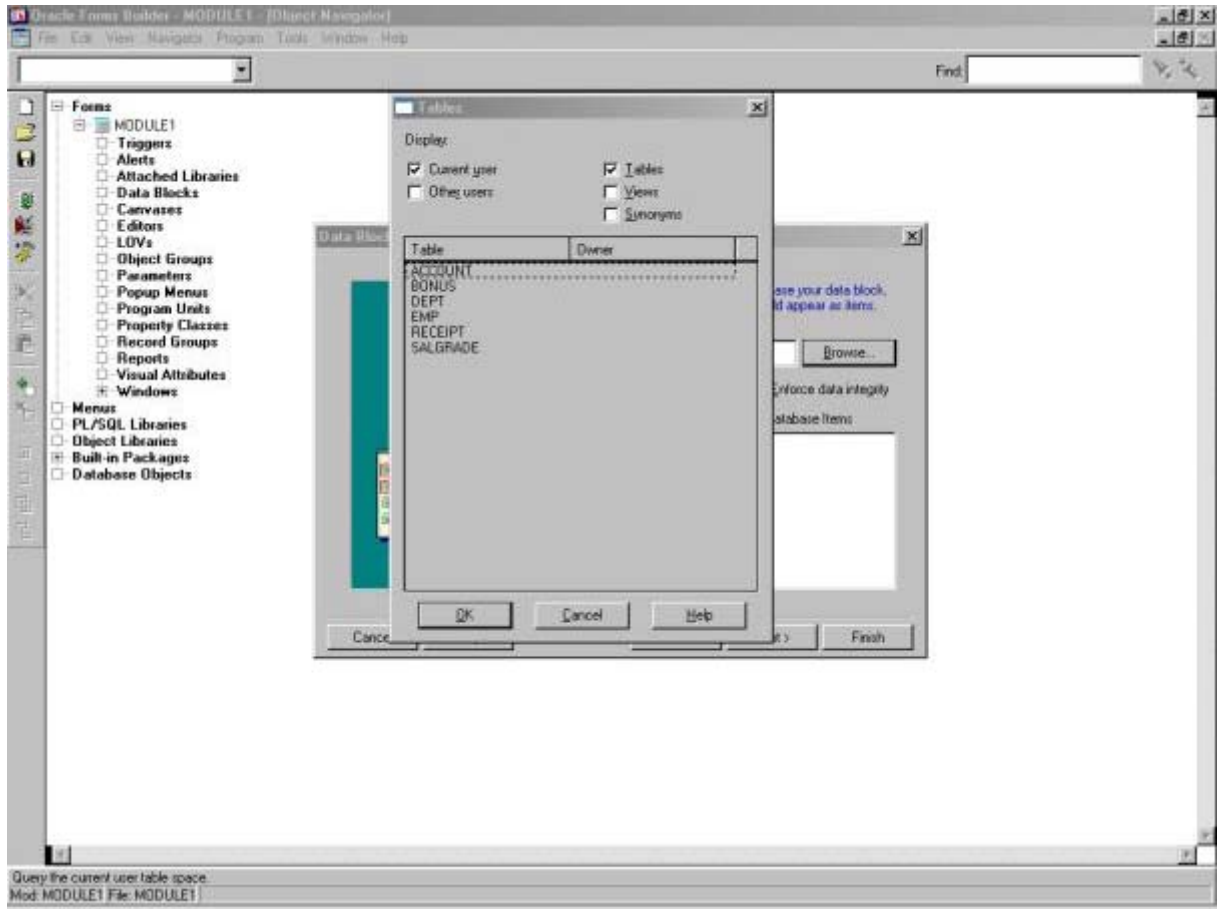


Şekil 9.1.1. Oracle Forms ile Form Hazırlama 1. Ekran

Bu ekrandan sırasıyla aşağıdaki seçenekler seçilerek işleme devam edilir.

**Data Blocks-> Use the data block wizard-> Table or View**

Bu ekranda kullanılacak tablolar tanımlanır. Tablolara ait bilgiler ekrana gelmeden önce oracle veritabanına erişmek için bir ekran gelir ve burada Browse denilerek oracle veritabanına erişebilmek için bağlantı yapılacak veritabanına ait kullanıcı adı ve şifresi yazılır. Kullanıcı adı olarak "scott", şifre olarak ise "tiger" yazılarak veritabanına bağlanılır. Bundan sonra ekrana bağlantı yapılan veritabanına ait tabloların listesi gelir. Bu listeden form hazırlanacak tablo seçilir.

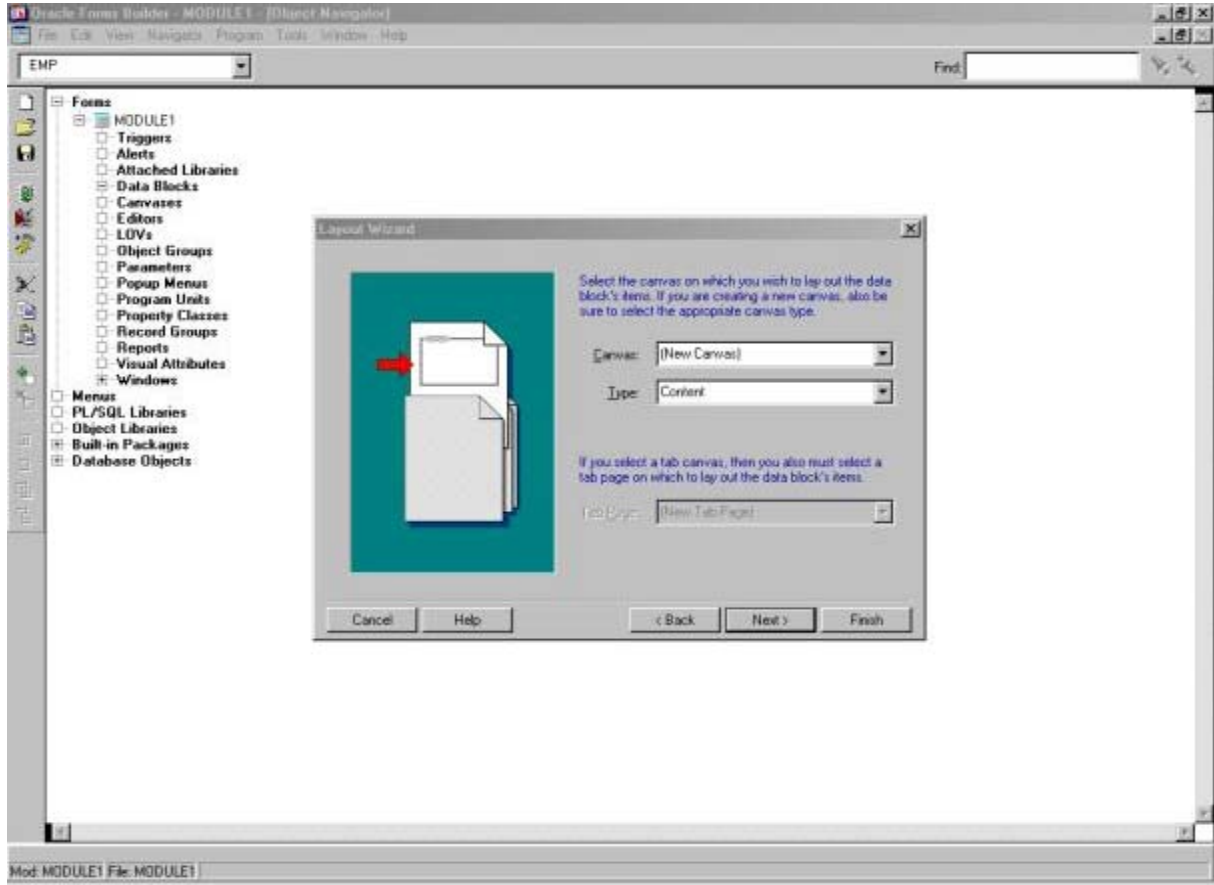


**Şekil 9.1.2.** Oracle Forms ile Form Hazırlama 2. Ekran

Burada EMP tablosunu seçip bütün alanlar Database items alanına atılır. Bu yapıldıktan sonra, programa hangi tabloyu kullanacağımızı bildirmiş ve tanıtmış oluruz. Daha sonra formun ekranda nasıl görüntüleneceği, arka fon alanının nasıl olacağı, tablodaki hangi alanların görüntüleneceği, bu alanlar ile yapılan işlemler için command buttonlar ile neler yapılacağı gibi ayarlamalar ve düzenlemeler yapılır. Burası tıpkı diğer programlama dillerindeki form düzenleme ekranlarında olduğu gibidir. Tek fark database nesnelerine erişilmesinin ve tanımlanmasının daha kolay olmasıdır.

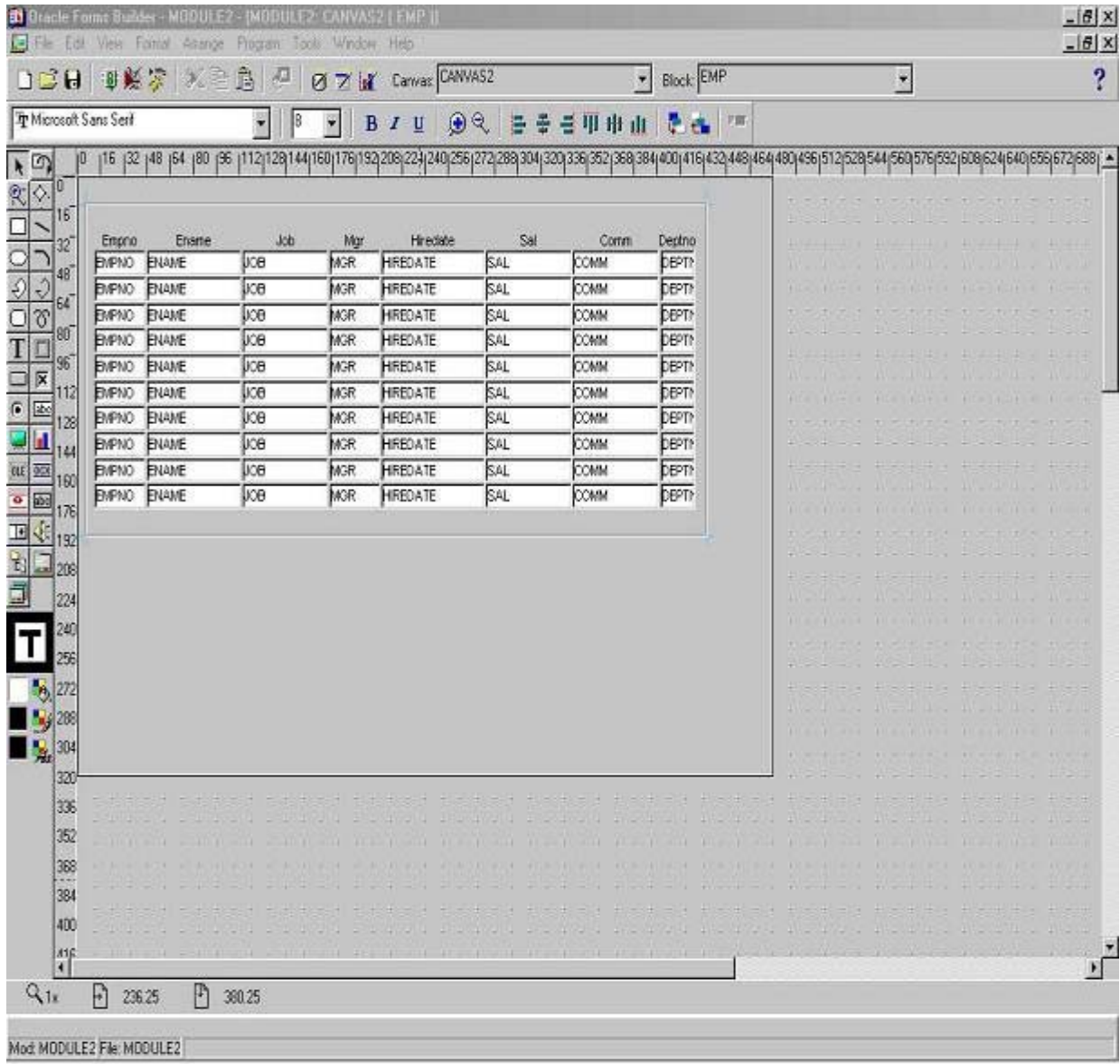
Çünkü veritabanının kendi program geliştirme programlarının veritabanına erişmesinde ayrıca bir API, ADO veya benzeri data nesnesi tanımlanmasına gerek olmayıp veritabanına bağlanılıp tablo seçildikten sonra nesnelere direk erişim mümkündür. Oysa Visual Basic veya diğer programlama dillerinde nesnelere erişim için data nesnesi tanımlanır ve her alan bu data nesnesine bağlanır. Bu aynı zamanda veritabanına bağlantı hızını yavaşlatarak programın program geliştirme araçlarına göre biraz daha yavaş çalışmasını sağlar. Zaten Oracle Forms gibi veritabanının kendi program geliştirme araçlarını kullanmanın avantajları da budur (hız ve veri nesnelere direk bağlantı gibi). Eğer program geliştirme araçları biliniyorsa program geliştirme araçları ile program yapmak; diğer programlama dilleri kullanılarak yapılan programlardan daha kolaydır.

Bu aşamada programın görüntüsü belirlenir.



**Şekil 9.1.3.** Oracle Forms ile Form Hazırlama 3. Ekran


New canvas seçili durumda iken next ile bir sonraki adıma geçilir ve buradaki Displayed Items bölümüne; seçilen tablodaki alanların formda kullanılacak olan alanları aktarılır ve next ile bir sonraki ekrana geçilir. Bu aktarma işlemi Access'deki form oluşturma ekranına çok benzemektedir. Bir sonraki adımda alanların ekranda nasıl görüntülenmesi gerektiği bildirilir. Burada satır veya sütunlu gibi seçenekler bulunur. Görünüş olarak, "Tabular" seçilip, Record displayed alanına 10 (Ekranda görüntülenecek kayıt sayısı) yazılır ve form hazırlanmış olur. Formun tasarım ekran görüntüsü aşağıdaki gibidir.

**Şekil 9.1.4.** Oracle Forms ile Form Hazırlama 4. Ekran (Tabular Ekran)

Bu ekrandan da diğer form hazırlama programlarında olduğu gibi form nesneleri ile ilgili tasarımlar ve ayarlamalar yapılabilir. Artık form hazır ve çalıştırılabilir. Formu çalıştırmak için trafik işaretine tıklanılması veya run seçeneğinin tıklanılması yeterlidir. Bu şekilde form çalıştırılırsa temel veritabanı işlemleri olan kayıt ekleme ve silme, düzeltme, arama işlemleri yapılabilir. Eğer form başka bir programlama dilinde (Visual Basic, C, C++, JAVA veya pascal(Delphi)) ile yapılsaydı; ekleme, silme gibi butonlar eklenip, her butonun event'ine kod yazmak zorunda kalınacaktı. Hazırlanan form programında butonlara gerek bulunmamaktadır. Temel veritabanı işlemleri olan kayıt ekleme ve silme, düzeltme, arama işlemlerini yapabilmek için çeşitli tuşlar kullanılmaktadır. Çalışan forms programında ana menüden help->keys tıklanarak veritabanı işlemlerinin hangi tuşlarla yapılabileceği öğrenilebilir.

Oracle form programında hazırlanan formlar tabular ise Record displayed alanına yazılan kayıt sayısı kadar bilgi ekranda görüntülenir. Programda bunun dışında tabular olmayan ve tablodaki her veriyi tek ekranda gösteren görünüş yapısı da vardır. Buna ait ekran görüntüsü aşağıdaki gibidir.

#### **Şekil 9.1.5. Oracle Forms ile Form Hazırlama 5. Ekran**

Hazırlanan form ekranında Select Query → Execute ile tablodan sorgu yapılarak bilgiler form üzerine getirilebilir.  ilgili data butonları kullanılarak ta datalar üzerinde gezinti yapılabilir.

Yukarıda anlatılan form master tablo ile yani tek tablodan oluşan bir formdur. Bunun dışında master-detail formlar da hazırlanabilir. Master-Detail formların özelliği birbiri ile ilişkili olan tablolardaki alanların görüntülenmesidir. Örneğin; bir müşteriye ait temel bilgiler görüntülenirken bu müşteriye yapılan satış bilgilerinin detayları aynı form üzerinde görüntülenebilir. Veya bir öğrencinin bilgileri ekranda görüntülenirken ekranın bir tarafındaki başka bir ekranda da öğrencinin harç bilgilerinin ve derslere ait sınav sonuçlarının görüntülenmesi gibi. Bu tip formların hazırlanmasında form sihirbazında master and detail blocks form seçilir.

Buna ait bir örnek ekran aşağıdaki gibidir. Birinci blok alanı master tablo diğer blok ise detail tablo alanındaki bilgileri görüntüler. İki tablo ilişkili olmak zorundadır.

#### **Şekil 9.1.6. Oracle Forms ile Form Hazırlama 6. Ekran**

Oracle Forms Programında bütün uyarılar İngilizce ve ekranın altında gösterilmektedir. Bunlar uyarı kutuları şeklinde ve Türkçe olarak hazırlanabilir ama bu olay başlı başına büyük bir program gerektirmektedir. İşlemler için butonlar eklenmelidir. Ayrı bir tablo kullanarak bir alana bilgi girilirken liste yardımı alınabilir. En üste görülen menü kendi icon ve eylemlerimiz için kullanılarak değiştirilebilir. Birkaç tane form varsa ve bu formlar arasında geçiş yapmak için ayrı veritabanında ayrı bir arayüz yapılabilir

Ama şunu bilmek gerekir ki, Forms builder ile form hazırlamak bu kadar basittir ama bazı işlemler için sihirbaz ile form hazırlamak yeterli değildir. Çünkü hazırlanan formları kullanacak operatörlerin tuş takımlarının görevlerini ezberlemeden işlemlerini butonlara ve menülere sahip profesyonel program ekranlarında daha kolay yapması gerekmektedir. Bunlarda menüler, butonlar ve ekrana gelecek listbox ve combobox gibi muhtelif ayrıntı ekranları ile mümkündür. Bunları hazırlamak için SQL ve PL-SQL iyi bilmek ve ilgili program geliştirme aracının eğitimini almak gerekmektedir.



## 10. FORMLAR ve VISUAL BASIC PROGRAMINDA FORM OLUŞTURMA

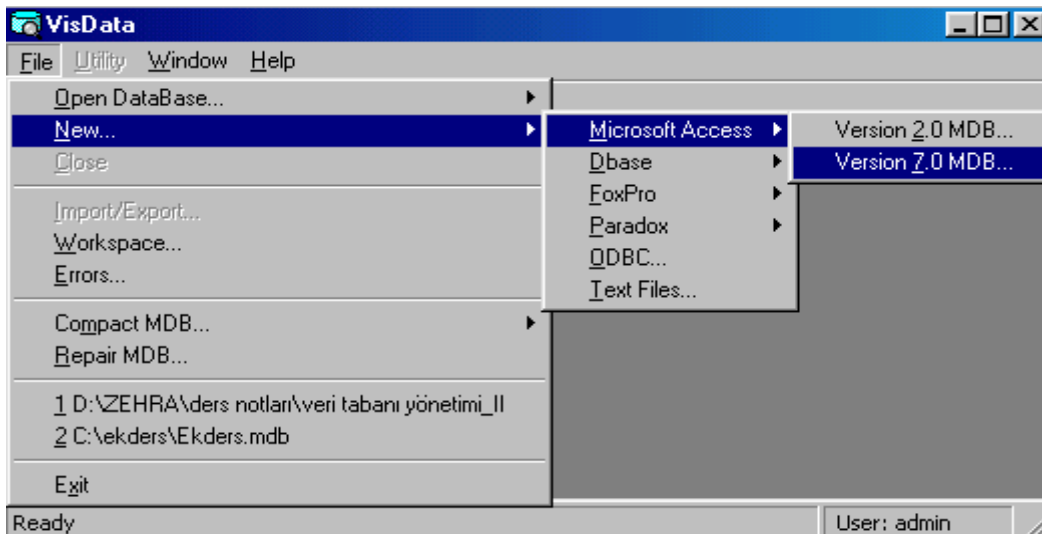
Bu bölümde Visula Basic 6.0 ile birlikte kullanılmaya başlayan veritabanı tasarım ve elemanları ile veri tabanı tasarımı ve form oluşturulması anlatılacaktır.

Visual Basic Programının 6.0 versiyonu ile kullanılmaya başlayan veritabanı tasarım ortamı çok kullanışlı olup program geliştiriciye çeşitli imkanlar sağlamaktadır. Gerek tasarım ve gerekse çalışma zamanında Visual Basic ortamından çıkmadan veritabanı dosyası tasarlamak, tablolar oluşturmak, form ve rapor tasarlamak çok kolay kolay ve oldukça zevkli hale getirilmiştir.

Veritabanı ile geliştirilen projelerin içerisine Visual Database bileşenleri dahil edilerek veritabanı projeleri geliştirilebilir, Oracle, SQL Server veya ODBC uyumlu diğer veritabanlarına erişim sağlanarak veritabanı formları ve raporları oluşturulabilir, sorgulamalar ve gerekli düzenlemeler yapılabilir. Query Designer programı ile de veritabanı dosyaları oluşturulabilir.

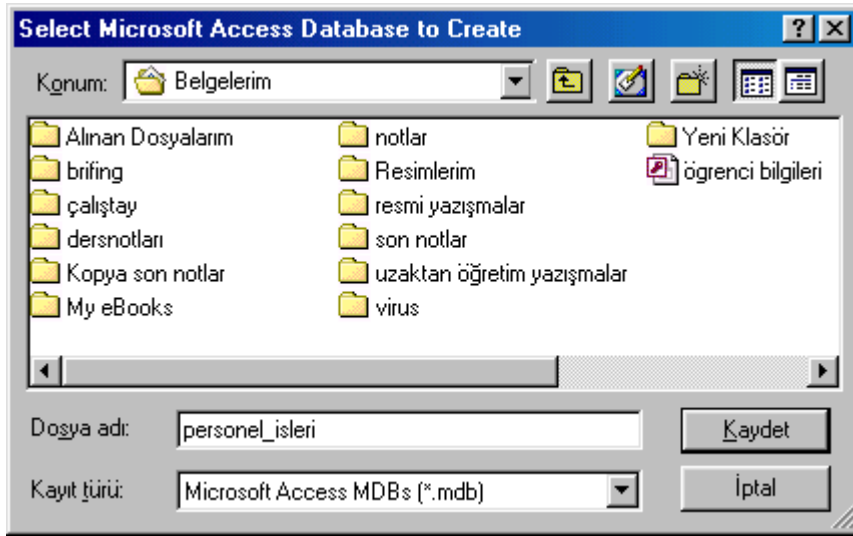
### 10.1 VERİTABANI TASARIMI VE VISUAL DATA MANAGER KULLANIMI

Access, Dbase, FoxPro, Paradox ve ODBC Veritabanı programlarının desteklediği diğer veritabanı dosyalarını oluşturmak için kullanılır. Visdata programı aracılığıyla çalışır. Programa geçmek için ana menüden “**Add-Ins-Visual Data Manager**” seçilir. New ile yeni veritabanı oluşturulur veya daha önceden hazır olan bir veritabanı **Open Database** seçeneği ile açılarak kullanılabilir.



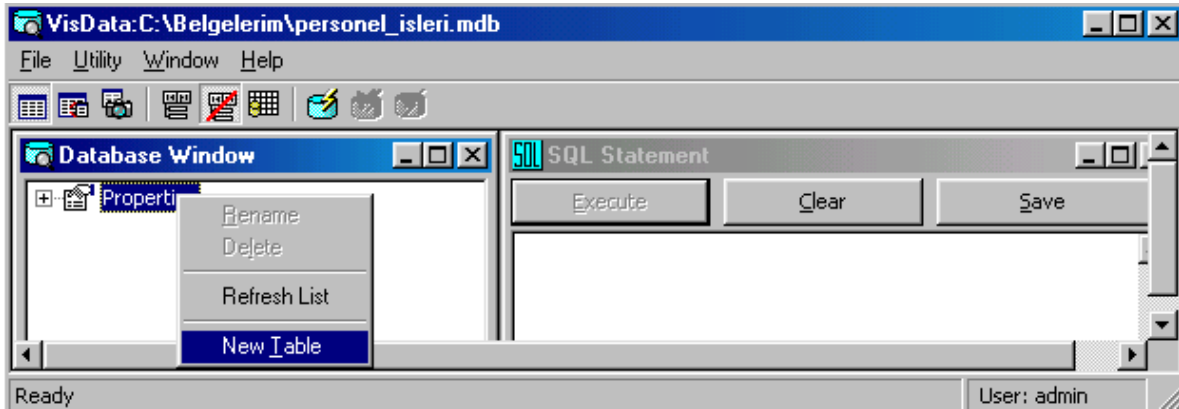
Şekil 10.1.1. VisData Program Ekranı-1





Şekil 10.1.2. VisData Program Ekranı-2

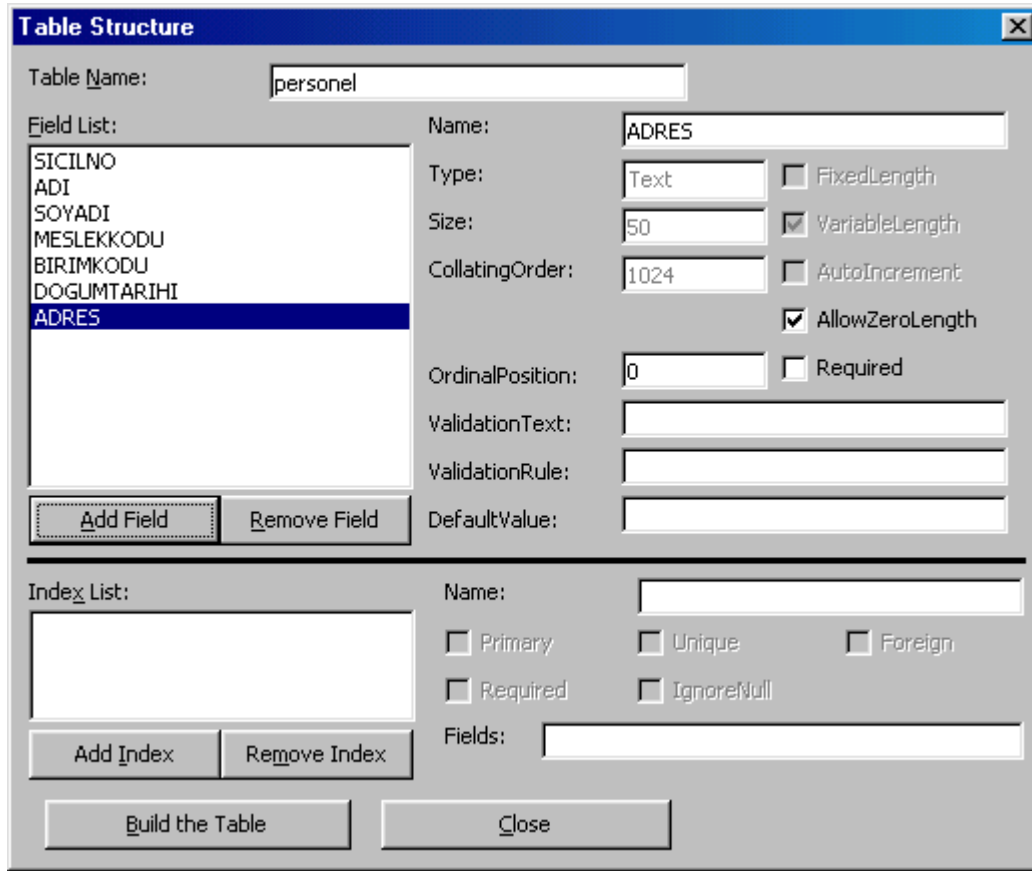
Bir isim verilerek kaydet butonuna tıklanılır ve Database yaratılır. Ekranı gelen görüntünün sol tarafında tablo işlemlerinin yapılabildiği Database Window ekranı, sağ tarafta ise SQL sorgulamalarının yapılabildiği SQL Statement ekranı bulunur.



Şekil 10.1.3. VeriTabanında Tablo Yaratılması Ekranı-1

Properties seçeneğinin üzerinde iken sağ mouse tıklanılarak **“New Table”** seçeneği ile yeni tablo yaratılır. Bundan sonra ekrana aşağıdaki veritabanı tasarım ekranı gelir. Burada tablonun isminin yazılacağı alan olan **“Table Name”** kısmına tablonun ismi olan Personel yazılır.

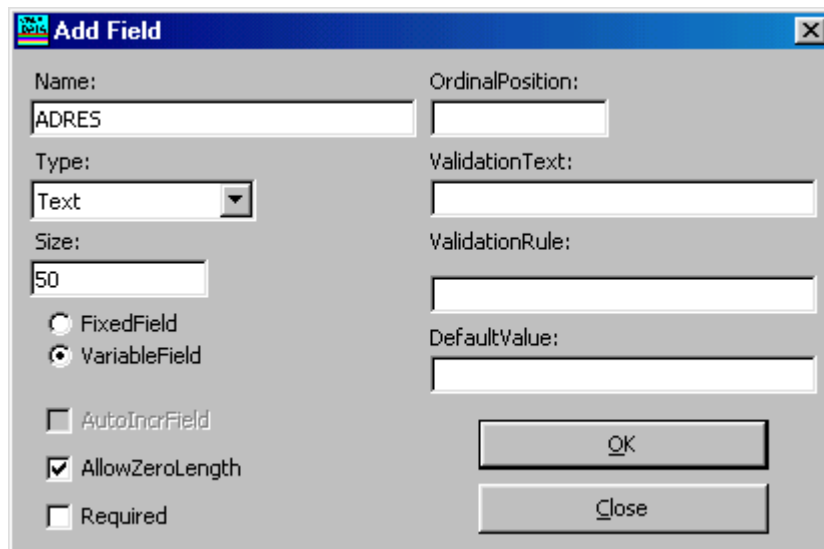
Bu ekranda ayrıca Add Field ile tabloya yani alanlar eklenebilir, Remove Fields ile tanımlanmış alanlar silinebilir. Alanlar için özel tanımlamalar ve kurallar tanımlanabilir veya default değerler oluşturulabilir. Tablo için Index yaratılabilir veya yaratılmış olan Index silinebilir. Tablo alanları için tabloya ait alanlar tanımlanır.



The 'Table Structure' dialog box is used to define the structure of a table. It features a 'Table Name' field set to 'personel'. The 'Field List' on the left contains fields: SICILNO, ADI, SOYADI, MESLEKKODU, BIRIMKODU, DOGUMTARIHI, and ADRES (which is selected). The right side shows the properties for the selected 'ADRES' field: Name (ADRES), Type (Text), Size (50), CollatingOrder (1024), OrdinalPosition (0), and various options like FixedLength, VariableLength, AutoIncrement, AllowZeroLength, and Required. Below the field list are 'Add Field' and 'Remove Field' buttons. At the bottom, there is an 'Index List' section with 'Add Index' and 'Remove Index' buttons, and a 'Build the Table' button. A 'Close' button is also present at the bottom right.

**Şekil 10.1.4.** VeriTabanında Tablo Yaratılması Ekranı-2

Add Field alanına tıklanılarak aşağıdaki ekranda tablo veri alanları tanımlanır.



The 'Add Field' dialog box is used to define a new field for the table. It contains fields for Name (ADRES), OrdinalPosition, Type (Text), Size (50), ValidationText, ValidationRule, and DefaultValue. There are also radio buttons for FixedField and VariableField (selected), and checkboxes for AutoIncField, AllowZeroLength (checked), and Required. 'OK' and 'Close' buttons are at the bottom right.

**Şekil 10.1.5.** Tablo Veri Alanlarının Tanımlanması Ekranı

Add Field alanlarının kullanımı ve açıklamaları aşağıdaki gibidir.

**Name** : Veri alanının ismi yazılır. SicilNo gibi.

**Type** : Veri alanının tipi yazılır. SicilNo için Integer, Adı için Text veya DoğumTarihi için Date/Time tanımlanması gibi.

**Size** : Veri alanı uzunluğu. Alan için tanımlanan uzunluk tipinden büyük değer yazılamaz.

**FixedField** : Alana yazılacak olan her kaydın uzunluğu sabit olarak saklanacaktır. Örneğin isim için 15 karakterlik alan tanımlanmış ise Ali içinde Abdurrahman içinde 15 karakterlik yer ayrılır.

**VariableField** : Seçili alanların uzunluğu farklı olabilir. Yazılan karakter sayısı kadar yer ayrılır.

**AutoIncField** : Sadece sayı alanları için kullanılan bir özelliktir ve sayıların değerini otomatik olarak birer artırır.

**AllowZeroLength** : Alana bilgi yazılması zorunlu değildir ve alan boş bırakılabilir.

**Required** : Alana bilgi yazılması zorunludur ve alan boş bırakılamaz.

**OrdinalPosition** : Veri alanının sırası yazılır. SicilNo alanının 1.sırada, Adı alanının 2.sırada yazılması gibi.

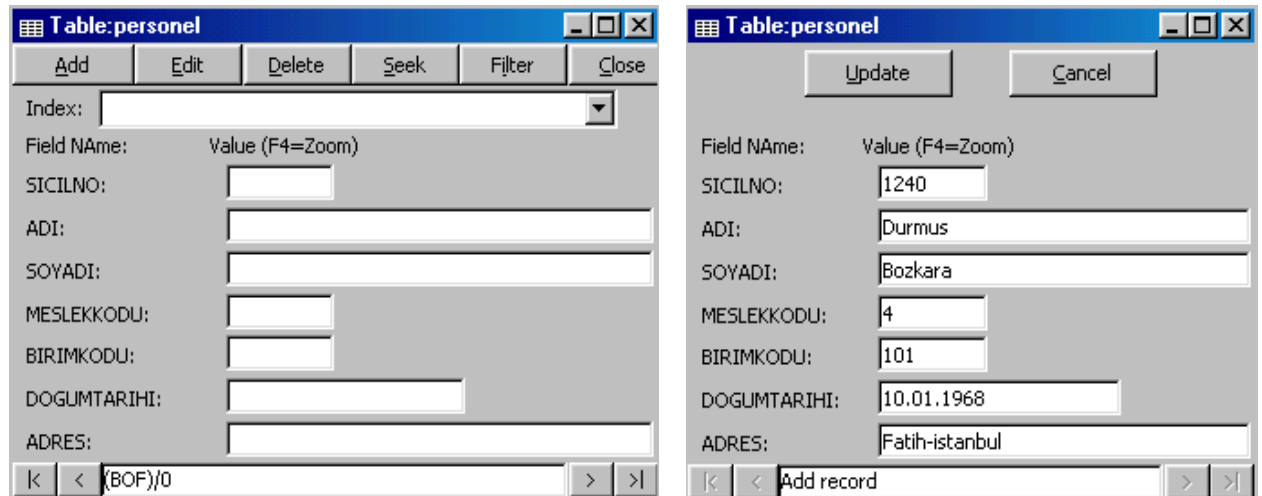
**Validation Text** : Alana Validation Rule kısmında tanımlanmış olan zorlayıcı (Constraint) dışında bir bilgi girildiğinde mesaj kutusu ile ekrana yazılacak mesaj.

**Validation Rule** : Zorlayıcının tanımlandığı alan.

**Default Value** : Alan için varsayılan değerdir. Örneğin, personelin hepsi XYZ bölümünde çalışıyorsa bölüm alanı için default değer XYZ yazılır ve veri girişi sırasında bu alan için XYZ bilgisi kendisinden geleceği için yeniden yazılmaz. İstenirse ekrana gelen default değer bilgi yazılışı sırasında değiştirilebilir.

Tüm alanların tanımlaması bittikten sonra **“Build the Table”** butonuna basılarak tablo kaydedilir ve VisData ortamına dönülür. Bu alanda örnek personel\_işleri projemize ait personel, maaşlar, çalışılan birim ve meslekler isimli 4 tabloyu tanımlayın.

Oluşturulan tabloya ait alanlara bilgi yazabilmek için DataBase Window penceresinde yer alan tablolardan istenileni sağ Mouse ile tıklanılarak ekrana gelen pencereden **“Open”** seçilir. Burada yapılabilecek işlemler aşağıdaki gibidir.



The image shows two screenshots of a software window titled "Table: personel". The left screenshot shows the "Add" button and the "Index" dropdown. The right screenshot shows the "Update" button and the "Add record" button. Both screenshots show a form with fields for SICILNO, ADI, SOYADI, MESLEKKODU, BIRIMKODU, DOGUMTARIHI, and ADRES.

Field Name	Value (F4=Zoom)
SICILNO:	
ADI:	
SOYADI:	
MESLEKKODU:	
BIRIMKODU:	
DOGUMTARIHI:	
ADRES:	


Left screenshot buttons: Add, Edit, Delete, Seek, Filter, Close. Right screenshot buttons: Update, Cancel.

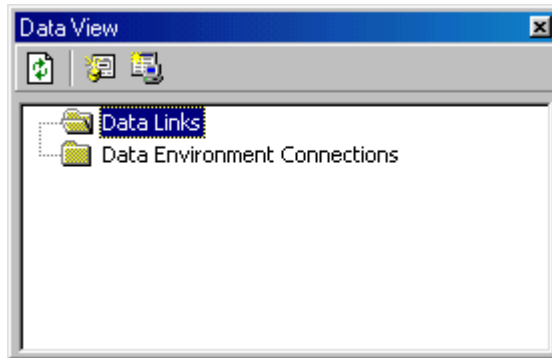
Şekil 10.1.6. Tablo Bilgi İşlemleri Ekranı

**Add** : Yeni kayıt eklenir.  
**Update** : Alanlara yazılan bilgiler güncellenir.  
**Delete** : Aktif kayıt silinir.  
**Find** : İstenilen kayıt bulunur.  
**Refresh** : Tablo alanları yeniden görüntülenir.  
**Close** : Bilgi giriş formu kapatılır.

## 10.2 VISUAL DATABASE BİLEŞENLERİ

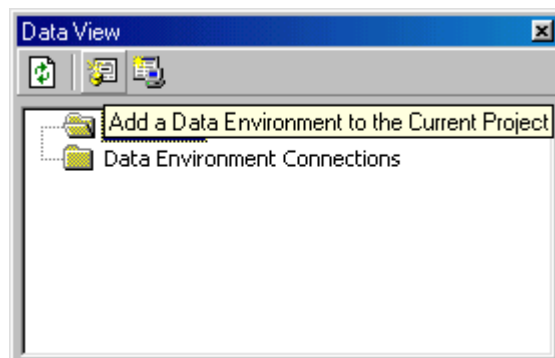
Visual Database elemanları aşağıdaki 4 bileşenlerden meydana gelmiştir.

**1. DataView Window** : Her proje ve bağlandığı veritabanı arasında görsel bir arabirim sağlar. DataView penceresini görüntülemek için Visual Basic ana menüsünden View -  Data View Window seçeneği kullanılır.



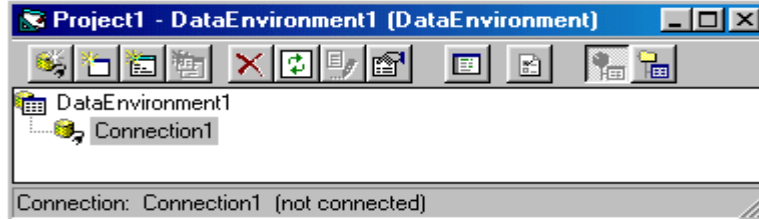
Şekil 10.2.1. DataView Window Ekranı-1

**2. Data Environment Designer** : Tasarım ile designer arasında etkileşimli bir arabirim sağlar. Veritabanı tablolarına bağlantı işleminin ilk aşamasıdır. DataView ekranındaki ikinci buton olan "Add a Data Environment to the current Project" 'e basılır.

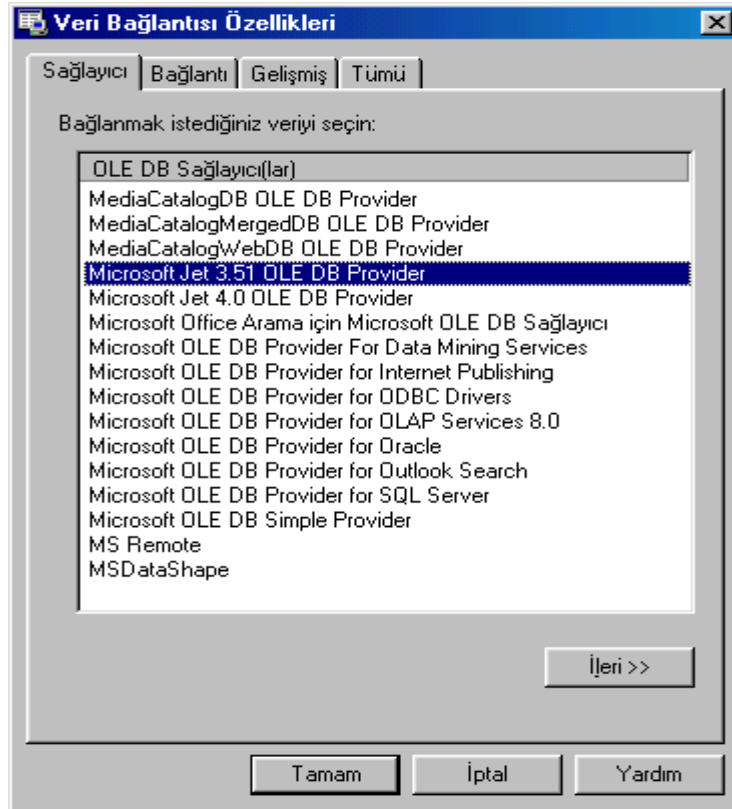


Şekil 10.2.2. DataView Window Ekranı-2

Burada aşağıdaki ekran olan veritabanı ve veritabanı tablolarına bağlantı işlemlerinin tanımlanması yapılır. Aşağıdaki ekranda bulunan Connection seçeneğine sağ Mouse ile tıklanılarak veri bağlantısı özellikleri” penceresi açılır ve açılan özellikler penceresinde tanımlamalar yapılır.

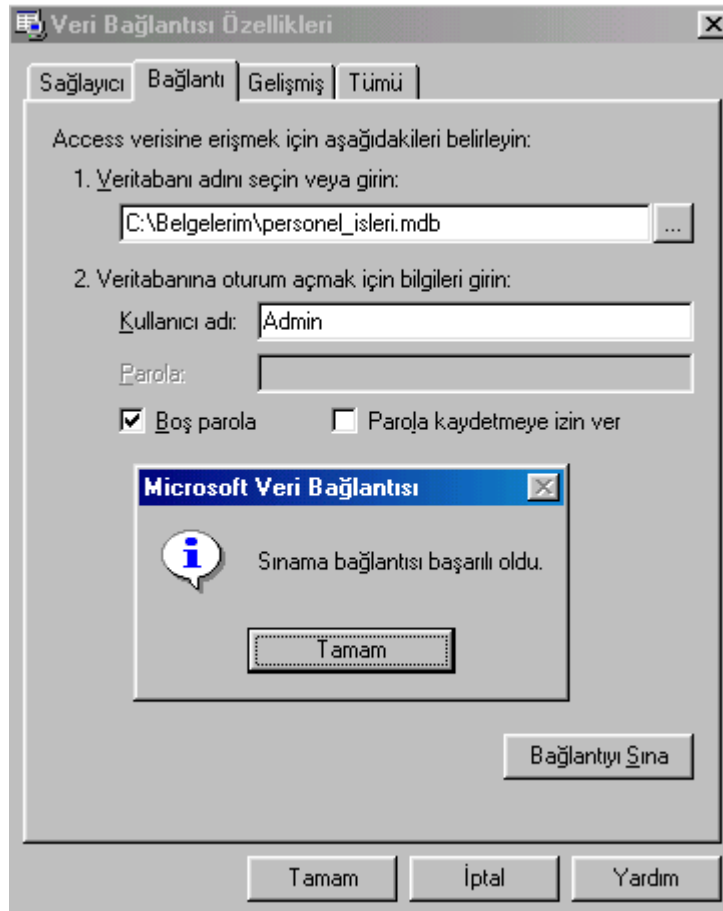


Şekil 10.2.3. Data Environment Ekranı



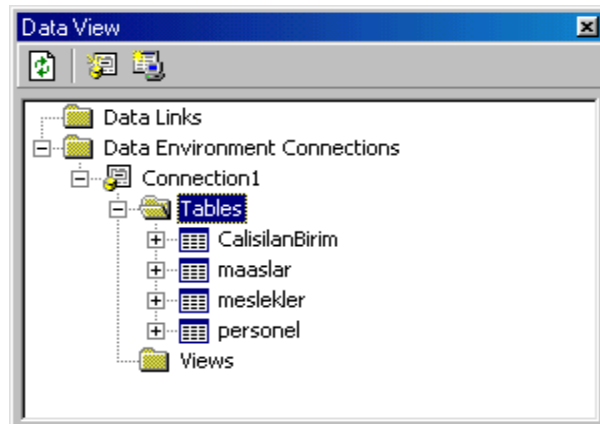
Şekil 10.2.4. Veri Bağlantısı Özellikleri Ekranı-1

Bu ekran 4 seçeneğe sahiptir ve kullanılan işletim sisteminin dil seçeneğine göre çalışır. Burada Windows İşletim sistemi Türkçe kullanıldığı için mesajlar Türkçe'dir. Kullanımı tıpkı sihirbaz ekranlarında olduğu gibi ileri (next) buttonlarına tıklanarak yapılır. Her ileri seçiminden sonra bir sonraki seçeneğe geçilir. Buradan da rahatlıkla görülebileceği gibi seçenekler arasında Oracle, SQL Server gibi veritabanlarına bağlantı seçeneği bulunmaktadır. Seçenekler içerisinde **“Microsoft Jet 3.51 OLE DB Provider”** seçilir ve kullanılacak veritabanını seçmek üzere ileri butonu ile bir sonraki adıma geçilir.



Şekil 10.2.5. Veri Bağlantısı Özellikleri Ekranı-2

İstenilen veritabanı dosyasının tam yolu girilir ve “Bağlantıyı Sına” (Test Connection) butonu ile bağlantının başarılı olup olmadığının testi yapılır. Bağlantının başarılı olduğuna dair mesajı aldıktan sonra tamam butonu ile bağlantı işlemi gerçekleştirilir. Bu işlemden sonra bağlantının sağlandığı veritabanına ait tablolar data view penceresinde yer alır.



Şekil 10.2.6. Veri Bağlantısı yapılmış Data View Ekranı

**Data View** penceresinde yer alan tablolardan istenilenler fareyle sürüklenmek suretiyle **Data Environment** penceresine bırakılır. Buraya bırakılan her tablo fare yardımıyla form üzerine alınabilir. Yani otomatik olarak veri alanları için text kutuları oluşturulabilir. Aynı zamanda **Data Report** üzerine bırakılarak otomatik rapor tasarımı da gerçekleştirilebilir.

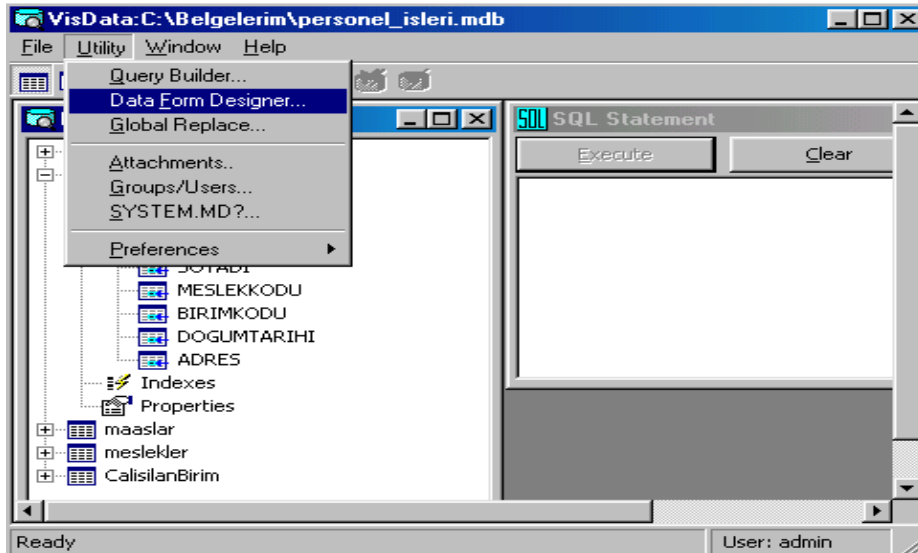
**3. Data Report Designer** : Rapor oluşturmada kullanılır.

**4. SQL Editör** : SQL işlemleri için kullanılır.

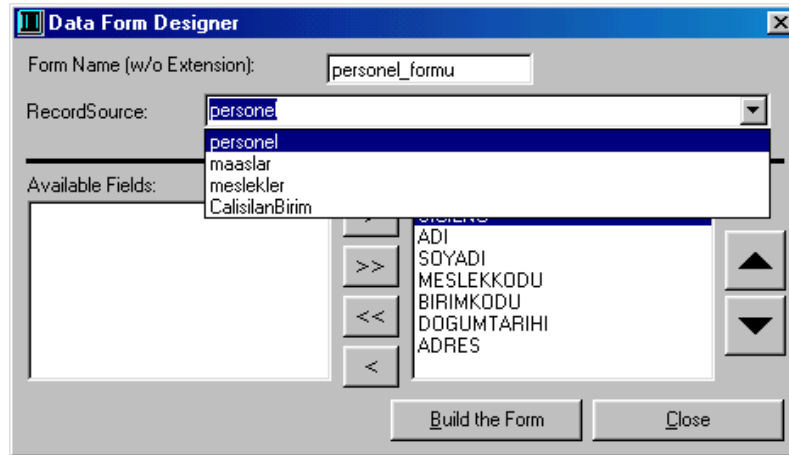
### 10.3 VISUAL BASIC ORTAMINA OTOMATİK VERİ FORMU

#### AKTARMAK

Visdata programı ile oluşturulan tablolardan otomatik olarak form sayısı hazırlanabilir. İşlemleri çok kolaylaştıran bu yöntemle program geliştirmek oldukça basittir. Bunun için VisData programından “**Utility-Data Form Designer**” seçilir.



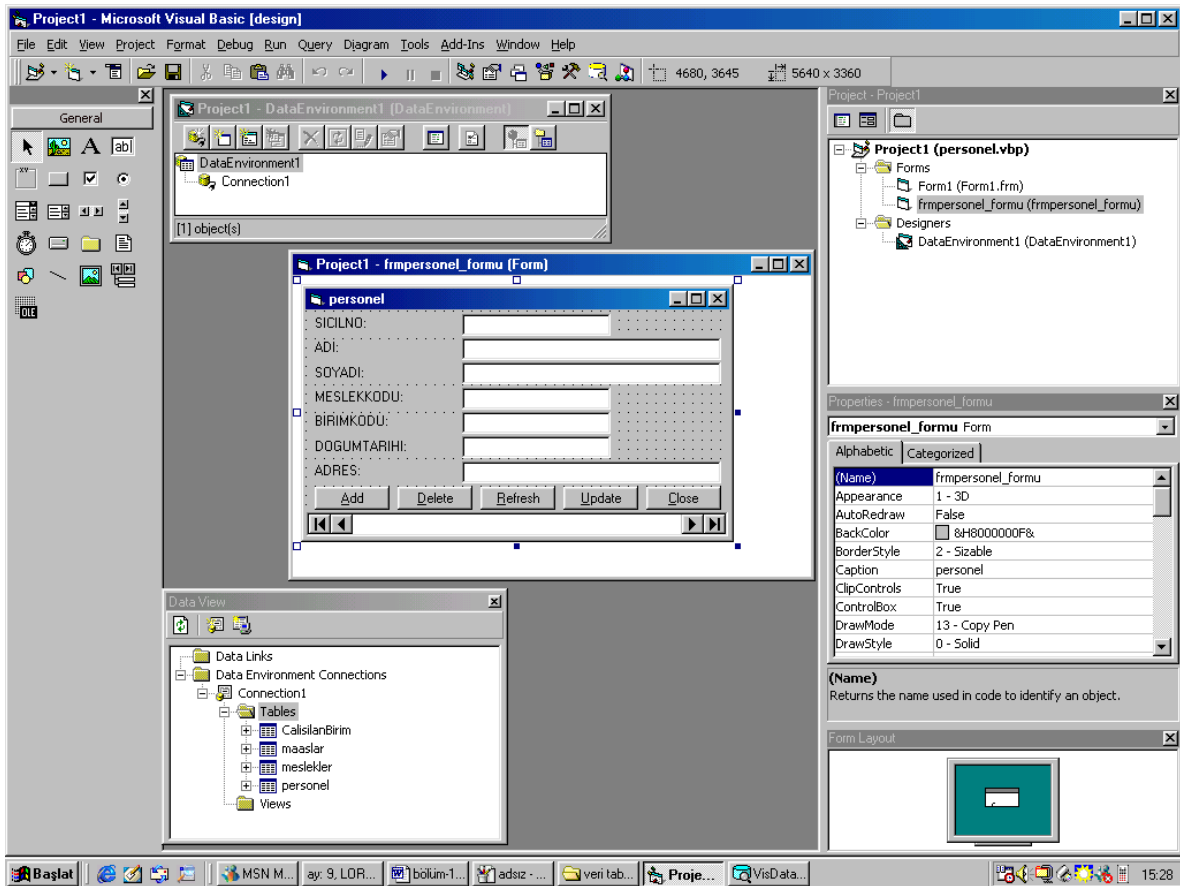
Şekil 10.3.1. Veri Bağlantısı yapılmış Data View Ekranı



Şekil 10.3.2. Data Form Designer Ekranı

Oluşturulan forma verilecek isim **Form Name** kısmına yazılır daha sonra kullanılacak tablo **RecordSource** alanından seçilir. Tabloya ait alanlar ekranın sol alt kısmında bulunan **Available Fields** alanına gelir. Formda listelenmesi istenilen alanlar buradan **Included Fields** alanına birer birer veya toplu olarak aktarılır. Build the Form butonu ile otomatik form oluşturma işlemine başlanılır ve kısa bir süre sonra form oluşturulur. Oluşturulan formda veritabanı işlemleri yapmak için hazır butonlar da bulunur. Ayrıca formun alt kısmında kayıtlar üzerinde hareket etmeyi sağlayan çubukta hazır olarak oluşur.

Form oluşturulduktan sonraki Visual Basic ekranı aşağıdaki gibidir.



Şekil 10.3.3. Data Form Designer ile Oluşturulan Form Ekranı-1

Form oluşturulduktan sonra Visual Basic ortamında program çalıştırıldığında ekrana artık normal bir program gibi tasarlanmış form gelir. Bu form ile formun üzerinde bulunan butonlar yardımı ile aşağıdaki işlemler yapılabilir.

**Add** : Yeni kayıtlar eklenir.

**Delete** : Aktif kayıt silinir.

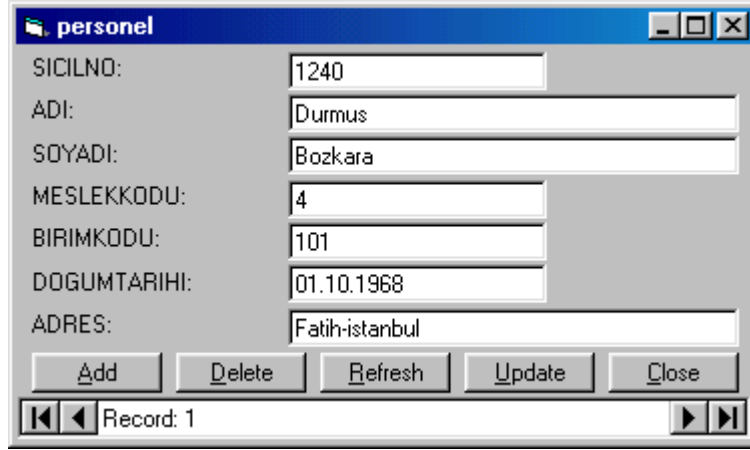
**Refresh** : Alan içeriklerinin görüntüsü yenilenir.

**Update** : Bilgiler üzerinde yapılan değişiklikler güncellenir.

**Kayıt Gezinti** düğmeleriyle kayıtlar arasında dolaşılabilir veya close ile programdan çıkılır.



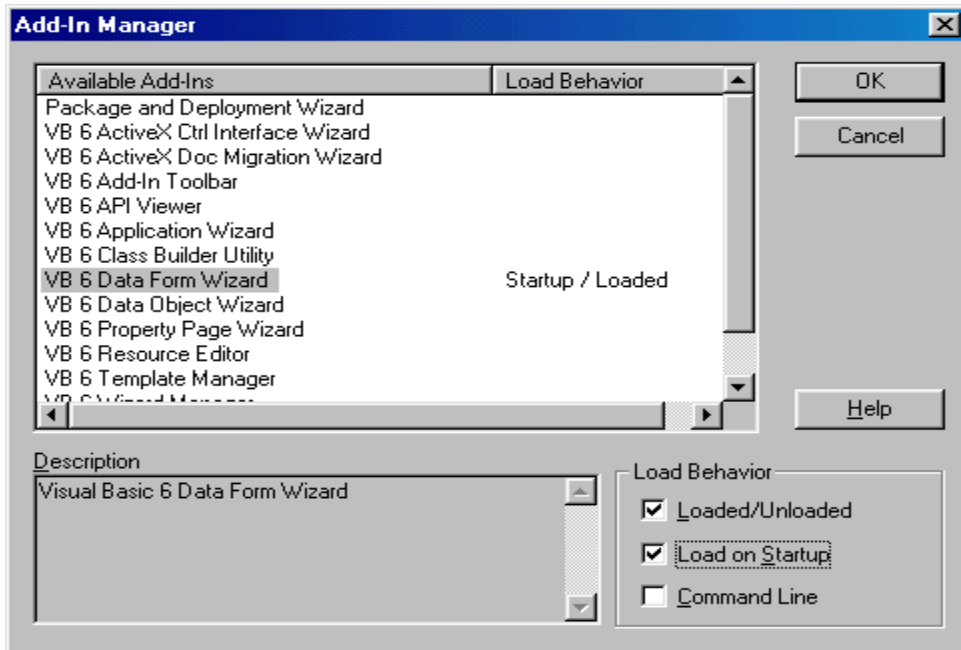
Oluşturulan form üzerinde Visual Basic tasarım ekranında istenilen düzenlemeler yapılabilir. Örneğin **Add** komut düğmesi seçilerek Caption özelliği **Yeni Kayıt** olarak değiştirilebilir.



Şekil 10.3.4. Data Form Designer ile Oluşturulan Form Ekranı-2

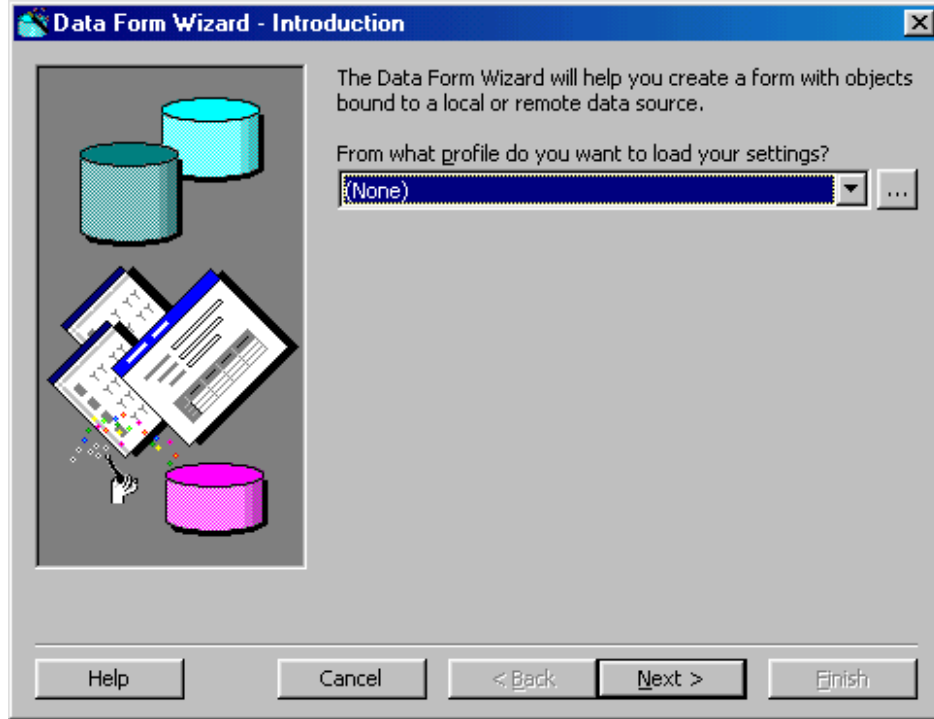
## 10.4 DATA FORM WIZARD İLE FORM OLUŞTURULMASI

Data Form Wizard (sihirbaz) ara birimi ile veritabanı arasında bir bağlantı kurularak tek-form, ana-alt form ve flexgrid içeren formlar kolayca hazırlanabilir. Data form Wizard ile form oluşturabilmek için Visual Basic ana menüsünden **Add-Ins - Data Form Wizard** seçilir. Bu seçenek bulunamazsa aynı menüdeki Add-In Manager dan seçerek yükleyebilir ve Visual Basic programı her çalıştırıldığında bu seçeneğinde otomatik olarak yüklenmesi sağlanabilir.

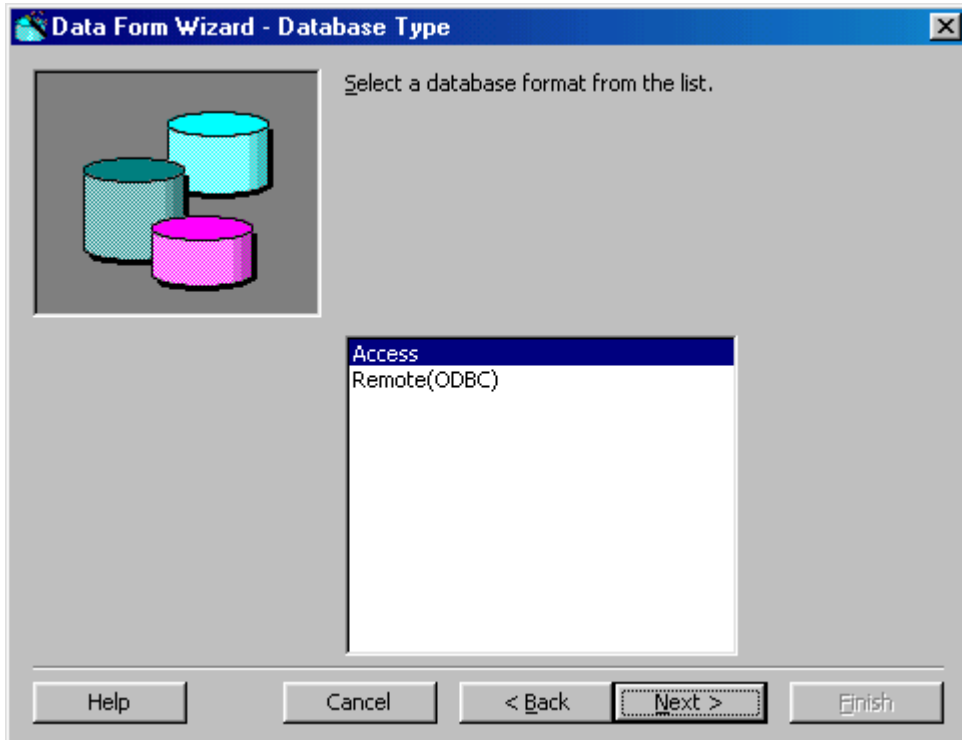


Şekil 10.4.1. Data Form Wizard Kullanıma Açılma Ekranı

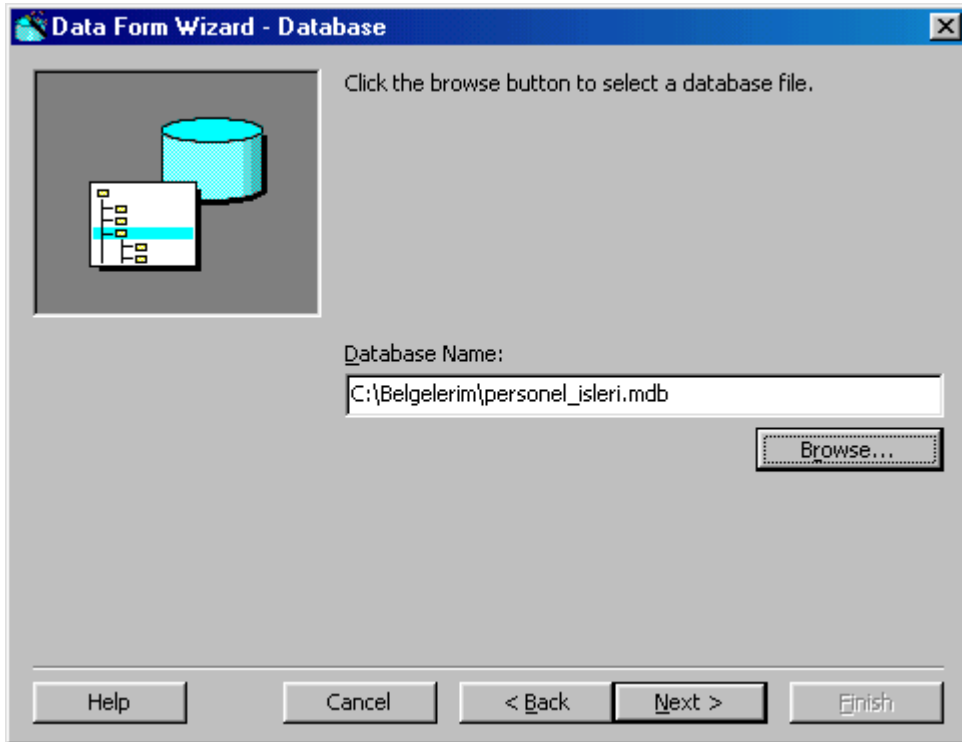
### 10.1.1. TEK TABLO İLE FORM OLUŞTURULMASI



Şekil 10.4.1.1. Data Form Wizard ile Form Oluşturulma Ekranı-1

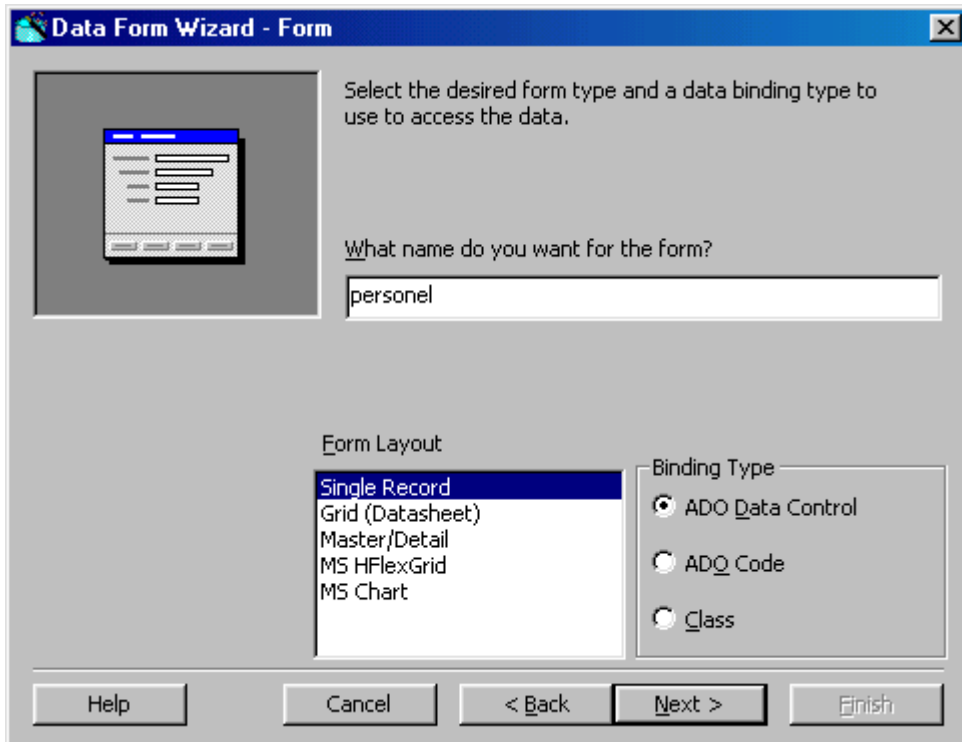


Şekil 10.4.1.2. Data Form Wizard ile Form Oluşturulma Ekranı-2



Şekil 10.4.1.3. Data Form Wizard ile Form Oluşturulma Ekranı-3

Veri tabanı ismi tam yol adı belirtilerek seçilir ve Next ile sonraki adımlara geçilir. Browse butonuna basılarak veritabanı istenilen alandan bulunarak seçilebilir.



Şekil 10.4.1.4. Data Form Wizard ile Form Oluşturulma Ekranı-4

Form Layout Listesindeki seçenekler ve özellikleri aşağıdaki gibidir;

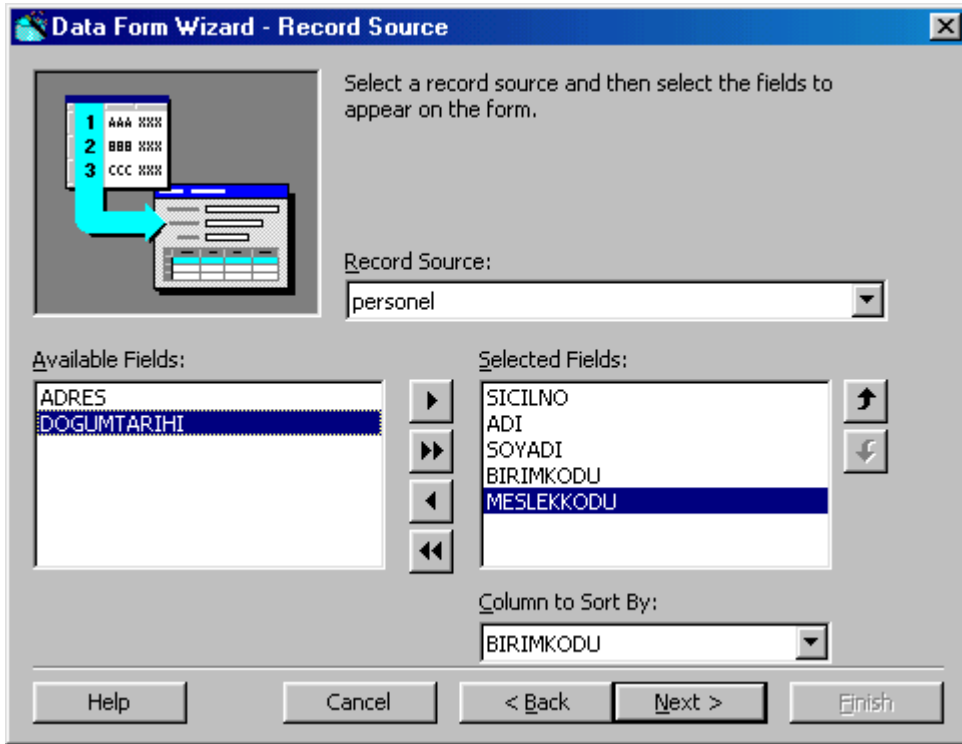
**Single Record** : Tek bir tabloya bağlantı sağlanarak form oluşturulur.

**Grid (Datasheet)** : Tek tablo üzerinde ve bu defa text kutuları yerine grid (ızgara) bileşeni kullanılır. Bununla bir çok kayıt aynı anda görülebilir ve üzerinde işlem yapılabilir.

**Master/Detail** : Ana/alt anlama gelen bu seçenekte birbiriyle ilişkili iki tablo kullanılarak ana ve alt formlar oluşturulabilir.

**MS Hflexgrid** : MsFlexGrid bileşeni kullanılarak form oluşturulur.

**Ms Chart** : Grafik bileşeni kullanılarak form oluşturulur.

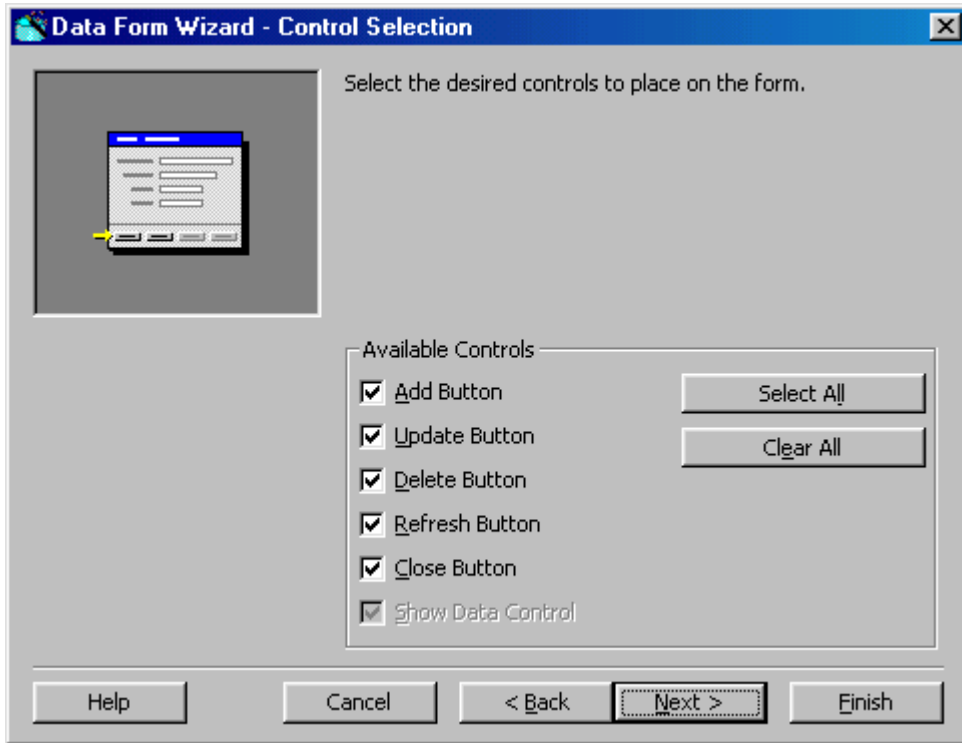


Şekil 10.4.1.5. Data Form Wizard ile Form Oluşturulma Ekranı-5

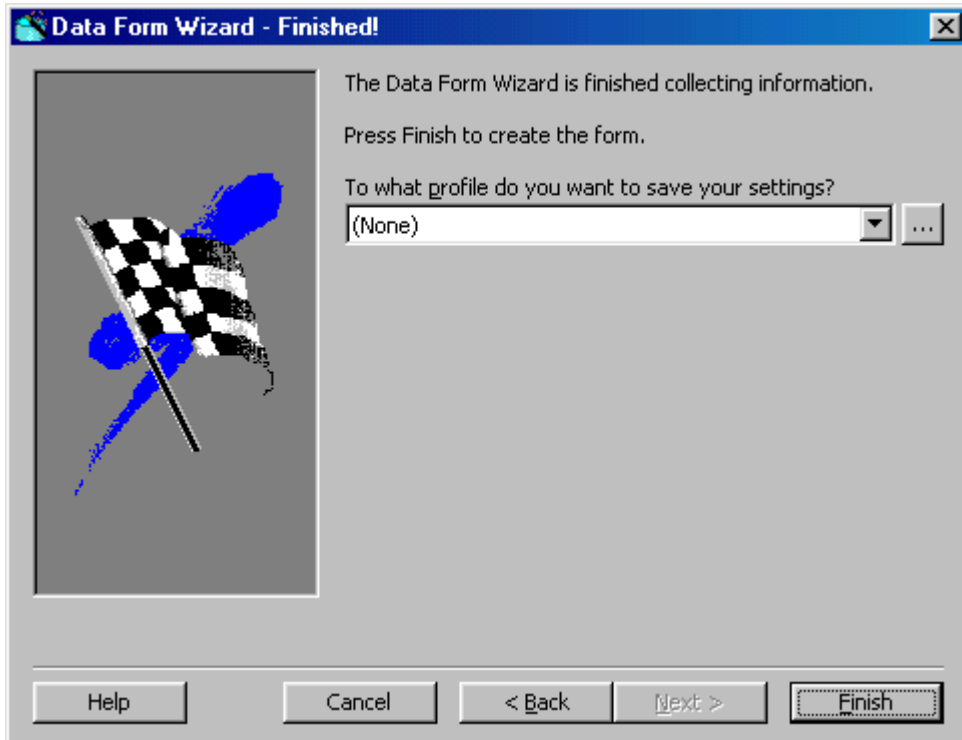
Record Source kutusunda bağlantı yapılacak tablo seçilir. Seçilen tabloya ait veri alanları Available Fields listesinde yer alır. İstenilen alanlar sağ taraftaki selected fields alanına aktarılır. Eğer istenirse Column to sort by kutusunda bir veri alanı seçilerek formda bilgilerin bu sıraya göre gelmesi sağlanabilir.

Örneğimizde personel tablosunda bulunan veri alanlarının tümü selected fields alanına aktarılmış ve veriler birim kodu alanına göre sıralanarak form oluşturulmuştur.

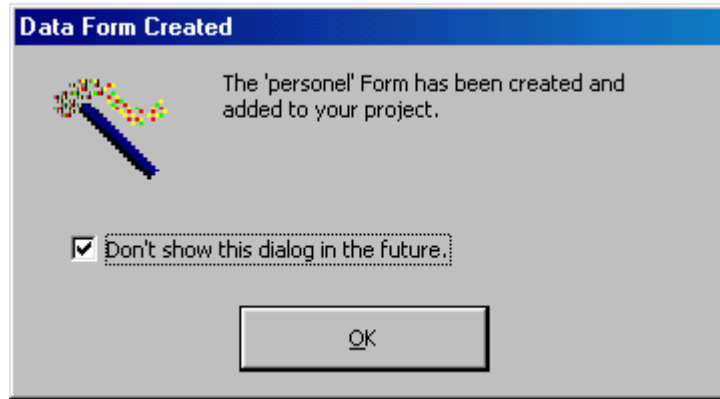
Bir sonraki adımda (sonraki şekil) form üzerinde bulundurulacak kontrol ve işlem butonları seçilir ve form üzerinde yer almaları sağlanır.



Şekil 10.4.1.6. Data Form Wizard ile Form Oluşturulma Ekranı-6

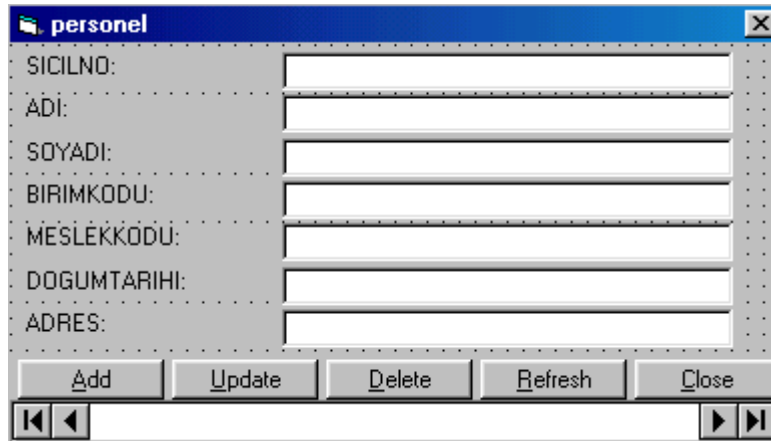


Şekil 10.4.1.7. Data Form Wizard ile Form Oluşturulma Ekranı-7



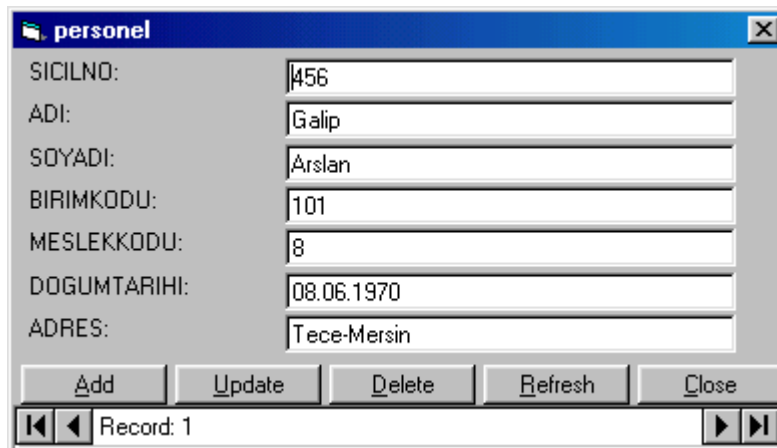
Şekil 10.4.1.8. Data Form Wizard ile Form Oluşturulma Ekranı-8

Sihirbaz işlemi tamamlandıktan sonra veritabanı içeren form hazırlanmıştır.



Şekil 10.4.1.9. Data Form Wizard ile Form Oluşturulma Ekranı-9

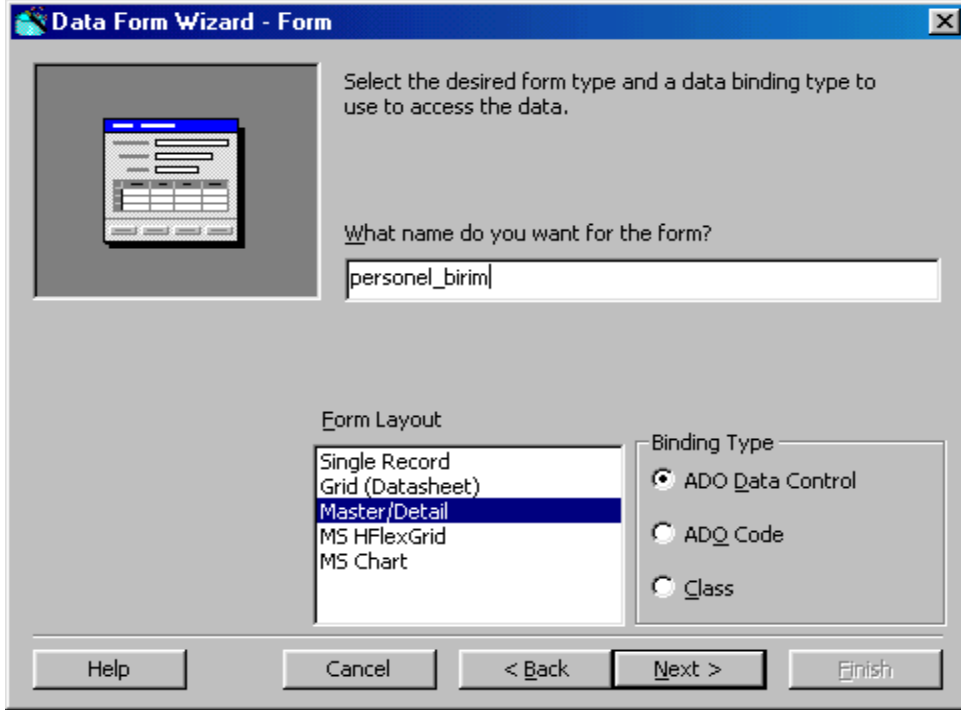
Hazırlanan formun program çalıştırıldıktan sonraki görüntüsü aşağıdaki gibidir.



Şekil 10.4.1.10. Data Form Wizard ile Form Oluşturulma Ekranı-10

### 10.1.2. MASTER / DETAIL İLE FORM OLUŞTURULMASI

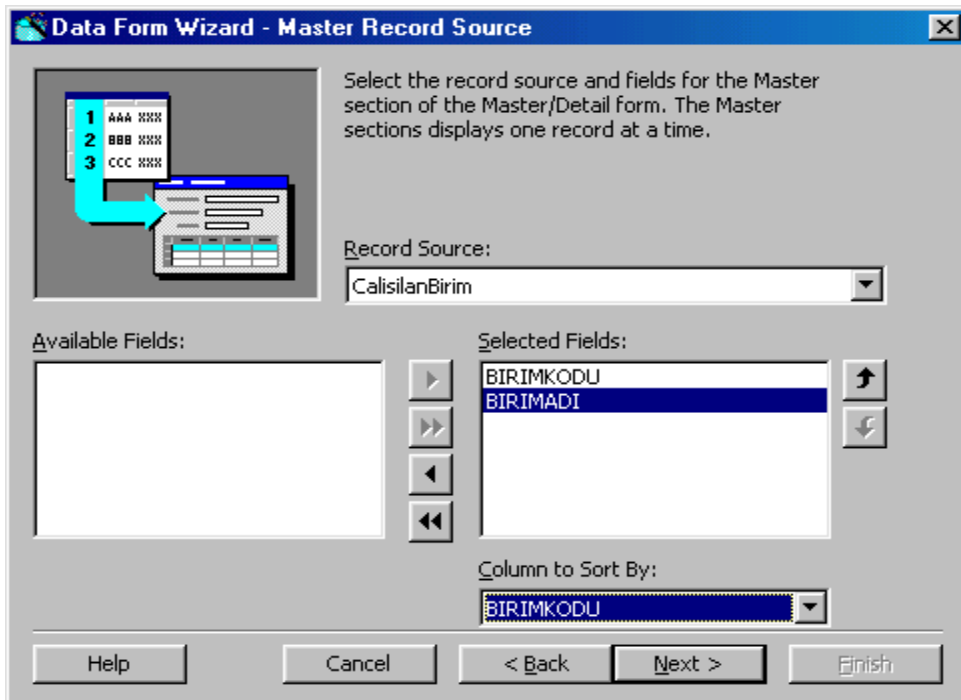
Daha önceki ilk 3 ekran single table ile form oluşturma ile aynıdır. Burada formun ismi yazıldıktan sonra Form layout kısmından Master/Details seçilir.



The dialog box titled "Data Form Wizard - Form" contains the following elements:

- Select the desired form type and a data binding type to use to access the data.**
- What name do you want for the form?** (Text input field containing "personel\_birim")
- Form Layout** (List box with options: Single Record, Grid (Datasheet), Master/Detail (selected), MS HFlexGrid, MS Chart)
- Binding Type** (Radio buttons: ADO Data Control (selected), ADO Code, Class)
- Buttons:** Help, Cancel, < Back, Next >, Finish

Şekil 10.4.2.1. Data Form Wizard ile Form Oluşturulma Ekranı-1

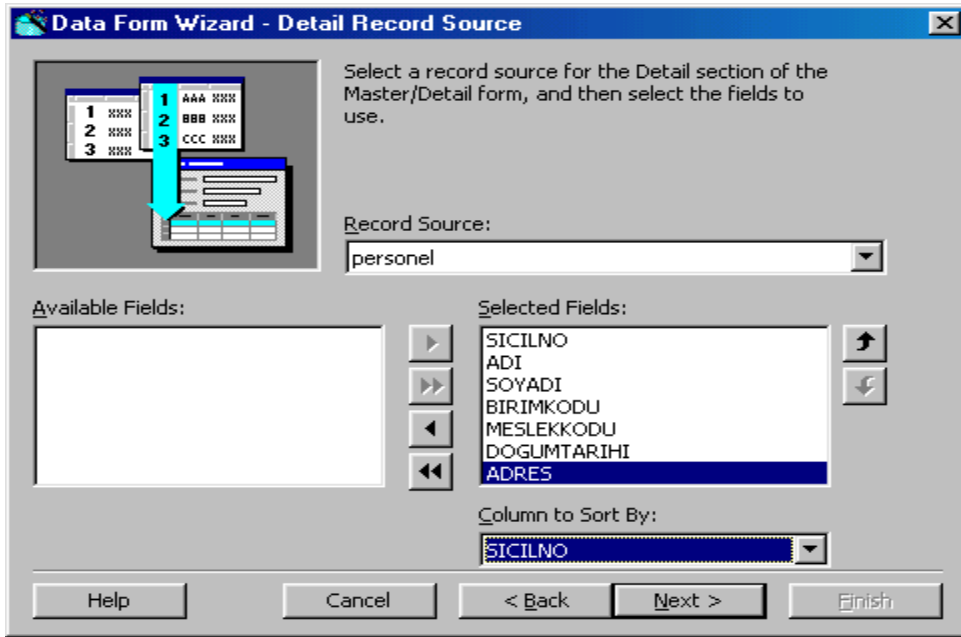


The dialog box titled "Data Form Wizard - Master Record Source" contains the following elements:

- Select the record source and fields for the Master section of the Master/Detail form. The Master sections displays one record at a time.**
- Record Source:** (List box containing "CalisilanBirim")
- Available Fields:** (Empty list box)
- Selected Fields:** (List box containing "BIRIMKODU" and "BIRIMADI")
- Column to Sort By:** (List box containing "BIRIMKODU")
- Buttons:** Help, Cancel, < Back, Next >, Finish

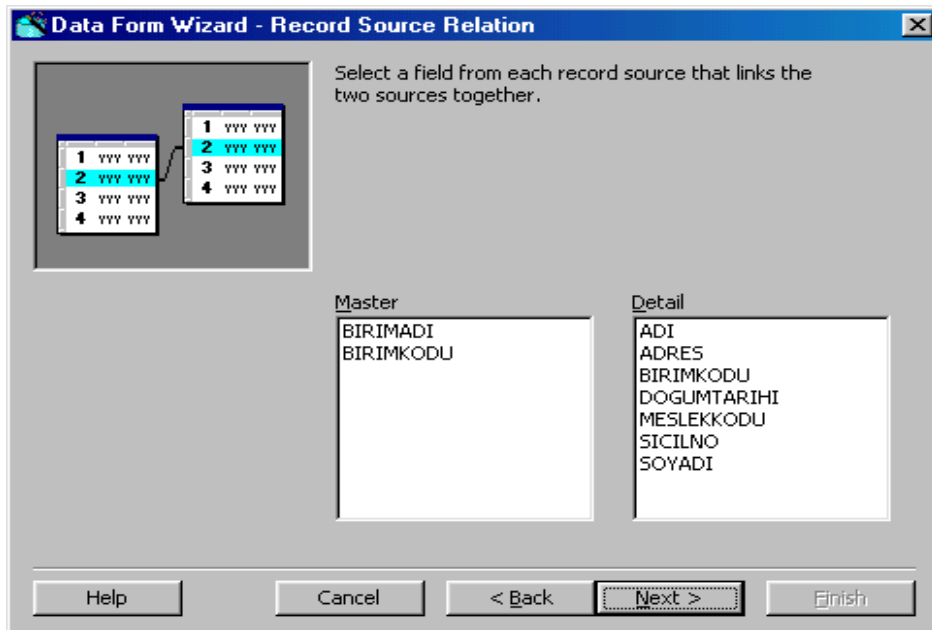
Şekil 10.4.2.2. Data Form Wizard ile Form Oluşturulma Ekranı-2

Burada birimler ve o birimde çalışan personellerin listesi veren ana ve alt formlar hazırlanmıştır. Öncelikle master tablo olan birim kodu tablosu ve listelenecek veri alanları seçilir. İstenilen bir veri alanına göre bilgiler sıralanabilir. Burada birim kodu alanına göre sıralama yapılması istenmiştir. Daha sonra detail form olacak tablo olan personel tablosu seçilir. Bu tablodaki tüm veri alanlarının listelenmesi istenmiştir. Burada en önemli konu; master ve detail olan tablolarda ortak bir alanın olması yani tabloların ilişkili olmasıdır. Örneğimizdeki iki tablo arasındaki ilişki birim kodu alanlarıdır.



The dialog box is titled "Data Form Wizard - Detail Record Source". It contains a preview of a master-detail form on the left. The main area has a "Record Source:" dropdown menu with "personel" selected. Below this are two lists: "Available Fields:" (empty) and "Selected Fields:" (containing SICILNO, ADI, SOYADI, BIRIMKODU, MESLEKKODU, DOGUMTARIHI, and ADRES). There are navigation buttons between the lists. At the bottom, there is a "Column to Sort By:" dropdown menu with "SICILNO" selected. The bottom of the dialog has buttons for "Help", "Cancel", "< Back", "Next >", and "Finish".

Şekil 10.4.2.3. Data Form Wizard ile Form Oluşturulma Ekranı-3

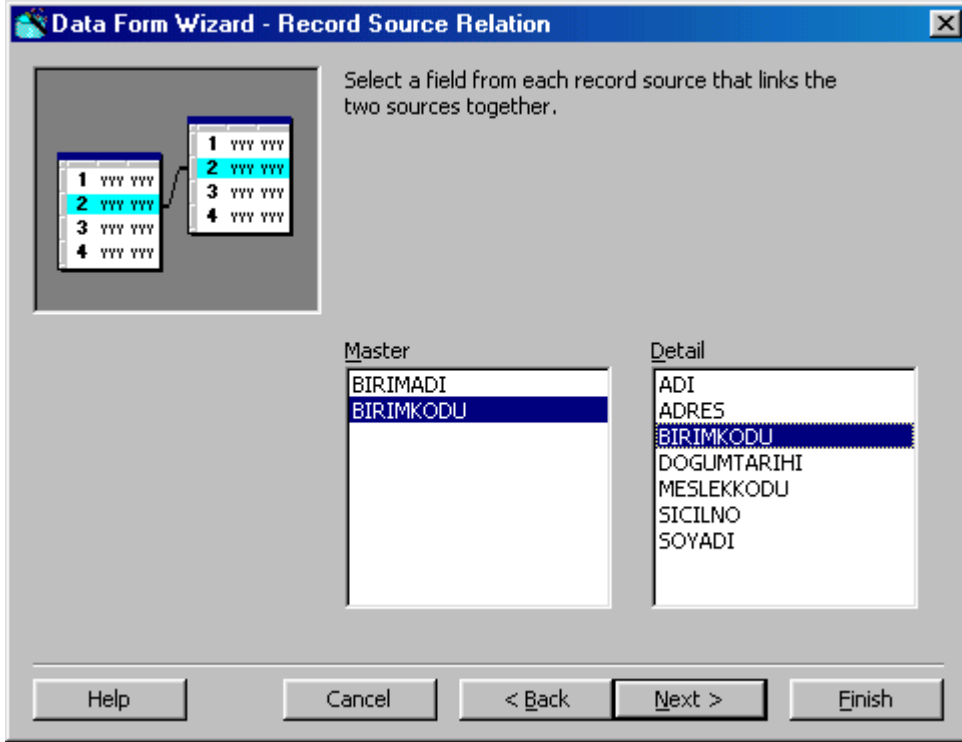


The dialog box is titled "Data Form Wizard - Record Source Relation". It contains a preview of a master-detail form on the left. The main area has two columns: "Master" and "Detail". The "Master" column contains "BIRIMADI" and "BIRIMKODU". The "Detail" column contains "ADI", "ADRES", "BIRIMKODU", "DOGUMTARIHI", "MESLEKKODU", "SICILNO", and "SOYADI". The "Next >" button is highlighted. The bottom of the dialog has buttons for "Help", "Cancel", "< Back", "Next >", and "Finish".

Şekil 10.4.2.4. Data Form Wizard ile Form Oluşturulma Ekranı-4




Buraya kadar master ve detail olarak kullanılacak tablolar ve bu tablolardan formun üzerinde görüntülenecek veri alanları tanımlanmıştır. Ekran görüntüsü Data Form Wizard ile Form Oluşturulma Ekranı-4 deki gibidir. Daha sonra bu ekrandan tablolar arasındaki ilişki olan ortak alanlar Mouse ile üzerine gelinerek işaretlenir. Örneğimizdeki bu alanlar birimkodu alanlarıdır.



The dialog box is titled "Data Form Wizard - Record Source Relation". It contains a visual representation of two tables on the left, each with 4 rows and 3 columns. The second row of both tables is highlighted. To the right, there is a text instruction: "Select a field from each record source that links the two sources together." Below this, there are two list boxes: "Master" and "Detail". The "Master" list box contains "BIRIMADI" and "BIRIMKODU", with "BIRIMKODU" selected. The "Detail" list box contains "ADI", "ADRES", "BIRIMKODU", "DOGUMTARIHI", "MESLEKKODU", "SICILNO", and "SOYADI", with "BIRIMKODU" selected. At the bottom, there are buttons for "Help", "Cancel", "< Back", "Next >", and "Finish".

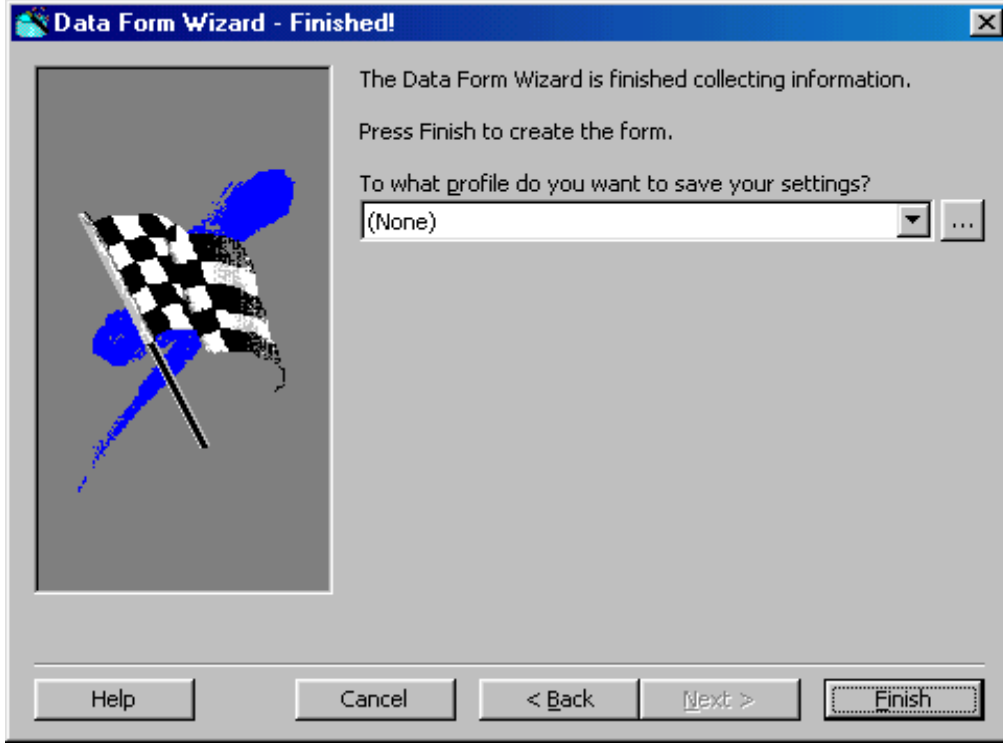
Şekil 10.4.2.5. Data Form Wizard ile Form Oluşturulma Ekranı-5



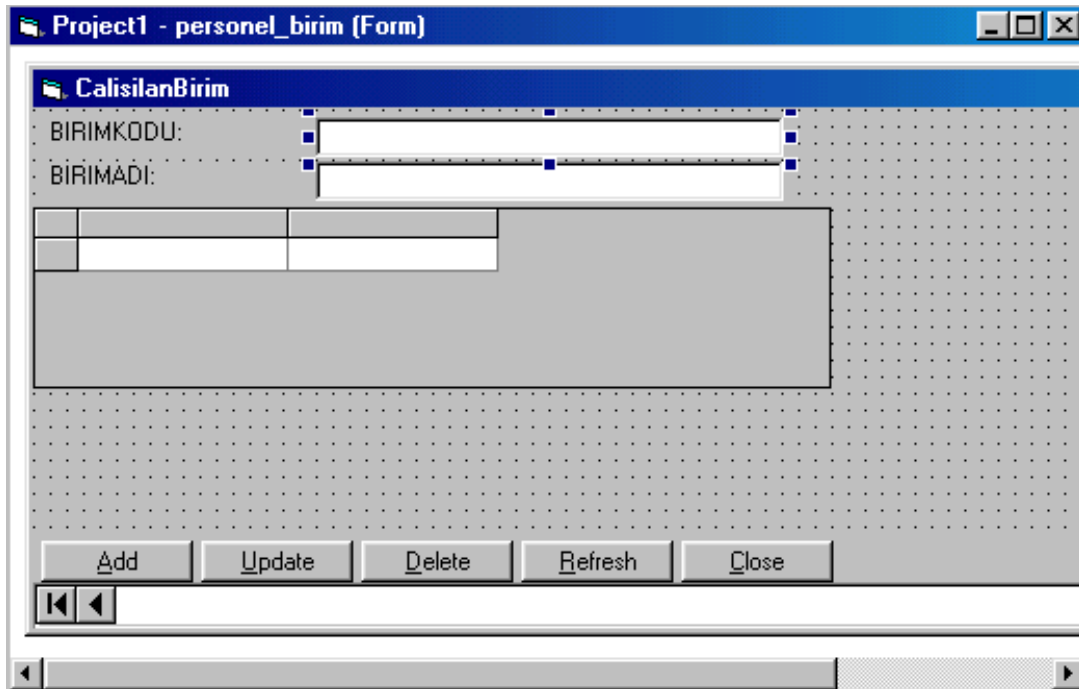
The dialog box is titled "Data Form Wizard - Control Selection". It contains a visual representation of a form on the left. To the right, there is a text instruction: "Select the desired controls to place on the form." Below this, there is a list of "Available Controls" with checkboxes: "Add Button", "Update Button", "Delete Button", "Refresh Button", "Close Button", and "Show Data Control". All checkboxes are checked. To the right of this list are two buttons: "Select All" and "Clear All". At the bottom, there are buttons for "Help", "Cancel", "< Back", "Next >", and "Finish".

Şekil 10.4.2.6. Data Form Wizard ile Form Oluşturulma Ekranı-6

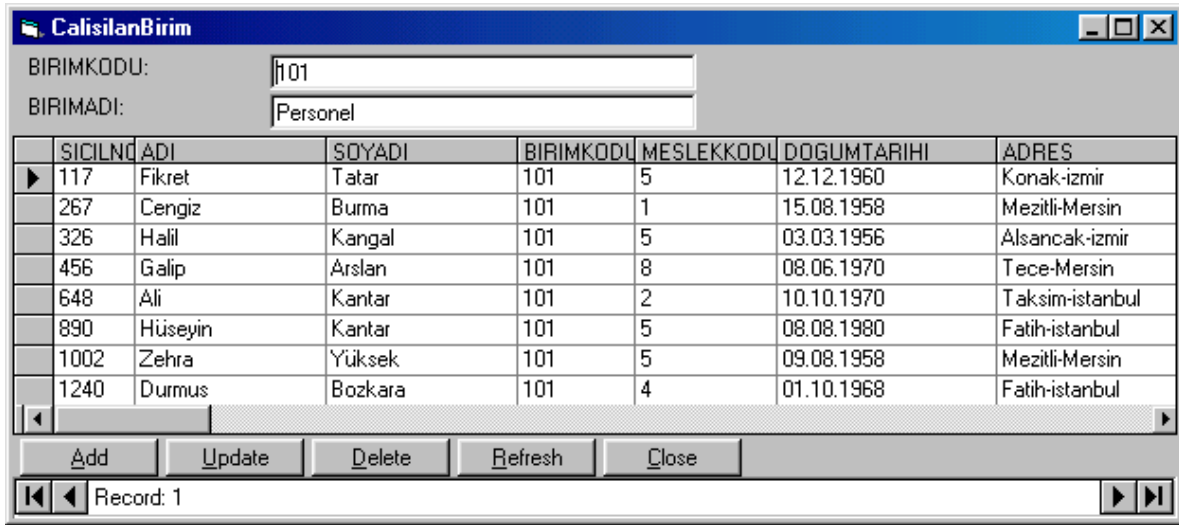
Form üzerinde olması istenilen butonlarda seçildikten sonra form tamamlanır.



Şekil 10.4.2.7. Data Form Wizard ile Form Oluşturulma Ekranı-7



Şekil 10.4.2.8. Data Form Wizard ile Form Oluşturulma Ekranı-8



**CalisilanBirim**

BIRIMKODU: 101

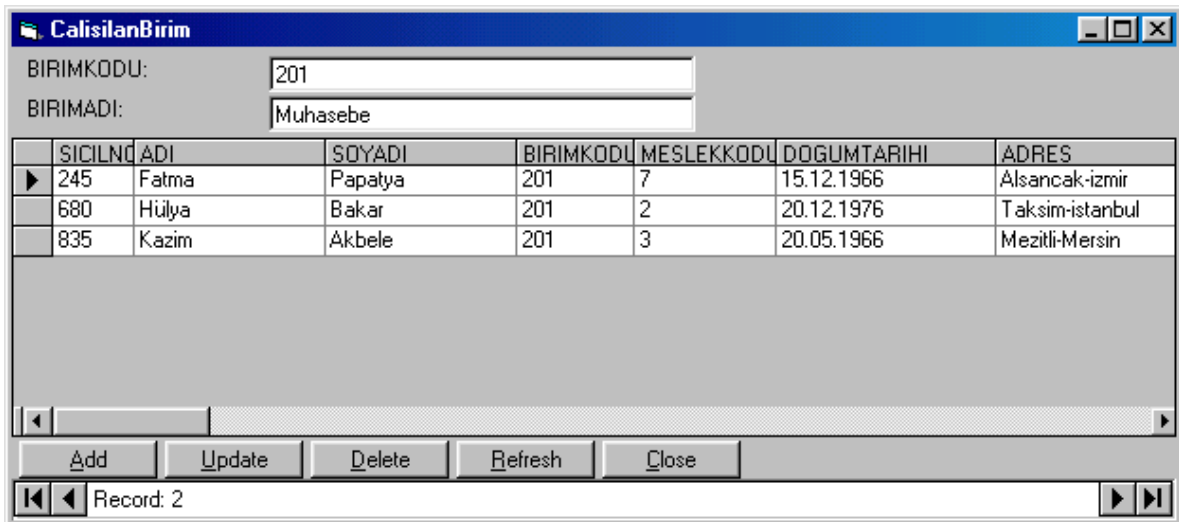
BIRIMADI: Personel

SICILNO	ADI	SOYADI	BIRIMKODU	MESLEKKODU	DOGUMTARIHI	ADRES
117	Fikret	Tatar	101	5	12.12.1960	Konak-izmir
267	Cengiz	Burma	101	1	15.08.1958	Mezitli-Mersin
326	Halil	Kangal	101	5	03.03.1956	Alsancak-izmir
456	Galip	Arslan	101	8	08.06.1970	Tece-Mersin
648	Ali	Kantar	101	2	10.10.1970	Taksim-istanbul
890	Hüseyin	Kantar	101	5	08.08.1980	Fatih-istanbul
1002	Zehra	Yüksek	101	5	09.08.1958	Mezitli-Mersin
1240	Durmus	Bozkara	101	4	01.10.1968	Fatih-istanbul

Add Update Delete Refresh Close

Record: 1

Şekil 10.4.2.9 Data Form Wizard ile Oluşturulan Form Ekranı Görüntüsü-1



**CalisilanBirim**

BIRIMKODU: 201

BIRIMADI: Muhasebe

SICILNO	ADI	SOYADI	BIRIMKODU	MESLEKKODU	DOGUMTARIHI	ADRES
245	Fatma	Papatya	201	7	15.12.1966	Alsancak-izmir
680	Hülya	Bakar	201	2	20.12.1976	Taksim-istanbul
835	Kazim	Akbele	201	3	20.05.1966	Mezitli-Mersin

Add Update Delete Refresh Close

Record: 2

Şekil 10.4.2.10 Data Form Wizard ile Oluşturulan Form Ekranı Görüntüsü-2

Görüldüğü gibi Visual Basic'te Data Form Wizard ile kullanışlı formlar hazırlamak oldukça kolaydır. Aynı yöntemle meslekler ve ilgili mesleklerde çalışan personeli görüntüleyen formlar hazırlanabilir. Veya personel tablosu master, maaşlar tablosu detail olacak şekilde çalışan personeli ve personele ait aylık maaşları görüntüleyen form hazırlanabilir.

## 11. RAPORLAR ve VISUAL BASIC PROGRAMINDA RAPOR OLUŞTURMA

Bu bölümde Visual Basic 6.0 ile birlikte kullanılmaya başlayan veritabanı tasarım elemanlarıyla rapor oluşturulması anlatılacaktır.

Rapor oluşturmak bir programın en önemli özelliklerinden biridir. İyi bir programı ön plana çıkaran özelliklerin en başında raporlamaların iyi ve çeşitli olması gelir.

### 11.1 DATA REPORT DESIGNER VE ÖZELLİKLERİ

Visual Basic teki raporlama özelliği Access'teki raporlama tekniğine oldukça benzemektedir. Rapor hazırlamak için Data Report Designer (DRD) kullanılır. Bu programla bir veritabanı kaynağına bağlantı kurularak değişik ve birbirleriyle ilişkili tablolara ait rapor oluşturmak oldukça kolay bir hale getirilmiştir.

Data Repor Designer birkaç önemli özelliğe sahiptir.

- Türkçesi sürükle ve bırak teknolojisi olan **Drag and Drop özelliği**. Bu özellik ile Data Environment Designer üzerindeki veri alanlarının sürüklenip bırakılmasıyla otomatik olarak text kutularını oluşturulmaktadır. Ve aynı zamanda sürüklenen alan yada alanlara ait DataMember ve DataField özellikleri de ayarlanmaktadır.
- **Toolbox kontrolleri** : Data Report Designer bileşeni kendisine ait olan bu toolbox elemanlarını kullanır. Bir data report designer elemanının projeye eklenmesiyle buna ait olan ve **DataReport** adını taşıyan kontrol elemanları toolbox'da yerlerini alırlar. Bunlar Label, shape, image, textbox ve line bileşenleridir. Ayrıca Function bileşeni, otomatik olarak Sum (toplama), Average (ortalama), Minimum (en küçük sayı) ve Maximum (en büyük sayı) işlemlerini desteklemektedir.
- **Print Preview özelliği** : Data Report Designer elemanının Show metodu kullanılarak veriler ön izleme penceresinde görüntülenebilir. Buradan direk olarak yazıcıya gönderilebilir. Bu özelliğin kullanılabilmesi için sistemde mutlaka bir yazıcının tanıtılmış olması gerekmektedir.
- **Print Reports özelliği** : Bir raporu programlama teknikleri ile yazdırmak için PrintReport metodu kullanılır.
- **File Export özelliği** : ExportReport metodu kullanılarak veriler HTML veya TEXT formatında kaydedilebilir.

## 11.2 DATA REPORT DESIGNER KONTROLLERİ

Rapor tasarımı kullanılabilecek 6 tane Data Report Designer elemanı vardır. Bunun haricindeki diğer kontrol elemanları rapor tasarımı kullanılamamaktadır.



**RptLabel** : Rapor başlıkları ve sabir ifadeler için kullanılan bir elemandır. Tasarım zamanında düzenlenebilir



**RptTextBox** : Çalışma zamanında ve sadece text formatındaki veri alanlarının içeriğini görüntüler. Tasarım zamanında bu bileşene veri girişi yapılamaz.



**RptImage** : Rapora resim eklemek için kullanılır. Bu bileşen aynı zamanda standart image kontrolüne de benzer. BMP, ICO, WMF, GIF ve JPEG resim formatlarını da destekler.



**PrtLine** : Rapor üzerinde yatay, dikey, çapraz vb. değişik çizgiler çizmek için kullanılır.



**RptShape** : Rapor üzerinde kare, dikdörtgen, çember, elips, oval, kare gibi değişik geometrik şekiller çizmek için kullanılır.



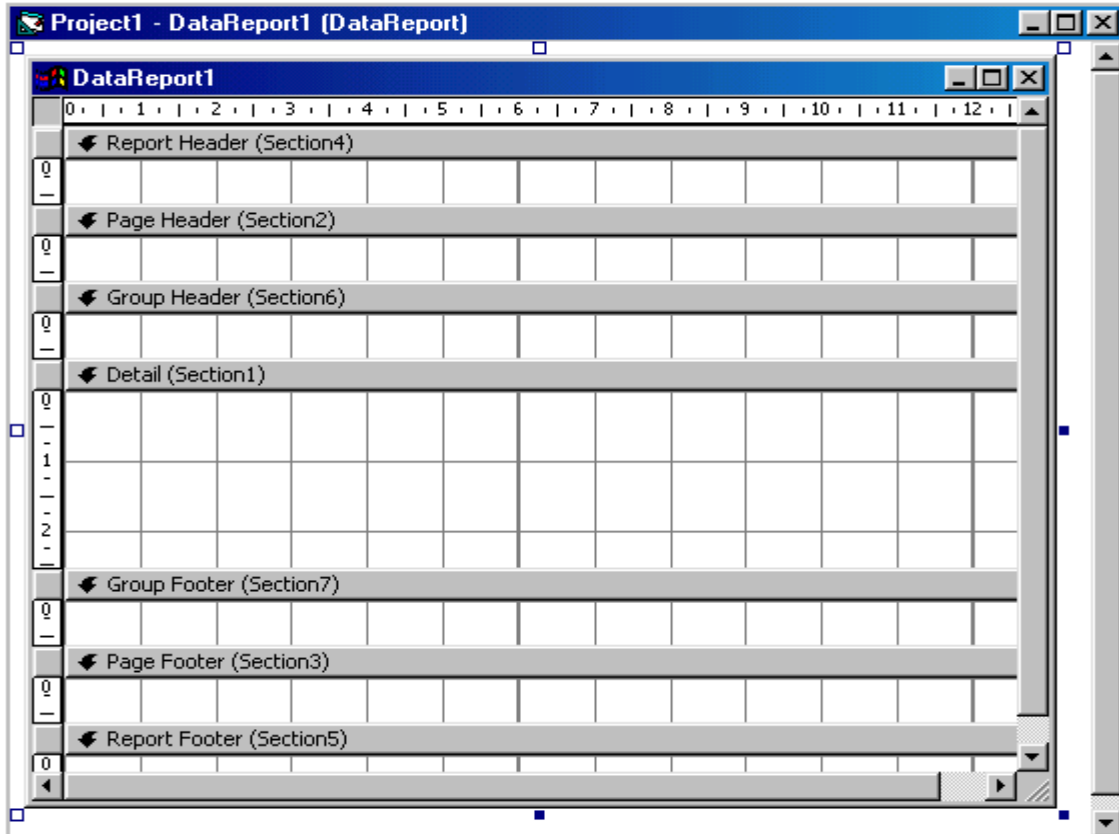
**PrtFunction** : Bu bileşen raporun sadece Footer bölümünde kullanılabilir. Bununla rapor üzerinde yer alan verilerin toplamı, ortalaması, en büyük ve en küçük değeri, kayıt sayısı hesaplanabilir.

## 11.3 DATA REPORT DESIGNER 'DA BİR RAPORUN YAPISI

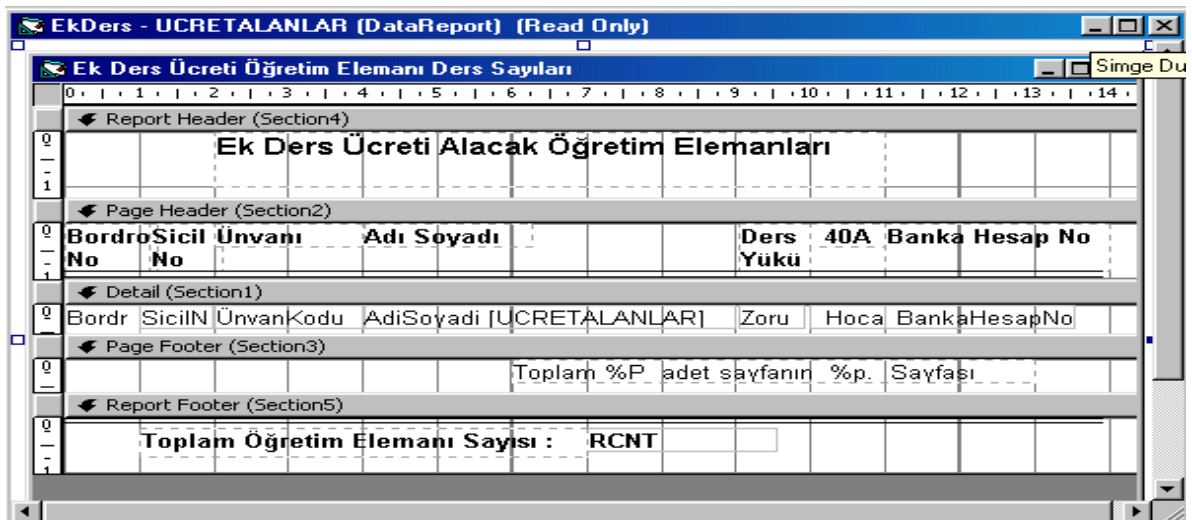
Bir Rapor aşağıda listelenen bölümlerden meydana gelir.

- Report Header (Rapor Başlığı) : Her raporun başlığında yazılacak olan ifadeler burada bulunur. Örneğin, rapor başlığı, yazar adı ve veritabanı dosyasının adı gibi.
- Page Header (Sayfa Başlığı) : Her sayfada yazılması istenilen ifadeler bu bölümde yer alır. Örneğin rapor başlığı gibi
- Group Header / Group Footer (grup başlığı / grup sonu) : Data Report'un tekrarlanan bölümlerini içerir. Her bir grup başı, grup sonu ile eşittir. Bu çift birlikte rapora eklenir veya kaldırılır.
- Details (gövde bölümü) : Raporun tekrarlanan bileşenleri buradadır. Veri alanları sürüklenip bu alana bırakılarak rapor oluşturulur. Tüm personele ait bilgiler bu bölümde listelenir.
- Page Footer (Sayfa sonu) : Rapora ait her sayfanın alt kısmında yazılacak olan bilgilerin bulunduğu bölümdür. Örneğin sayfa numarası, tarih bilgisi gibi

- Report footer (Rapor sonu) : Raporun sonunda yer alması gereken bilgiler burada yer alır. Örneğin, rapor özet bilgileri ve adres gibi. Report footer son sayfanın sayfa başlığı ve sayfa sonu arasında yer alır. Topla, ortalama gibi rptFunction bileşeni sadece bu bölümde yer alır.



Şekil 11.3.1. Data Report Designer Bölümleri Ekran Görüntüsü



Şekil 11.3.2. Data Report Designer ile oluşturulmuş Örnek Rapor Ekranı-1

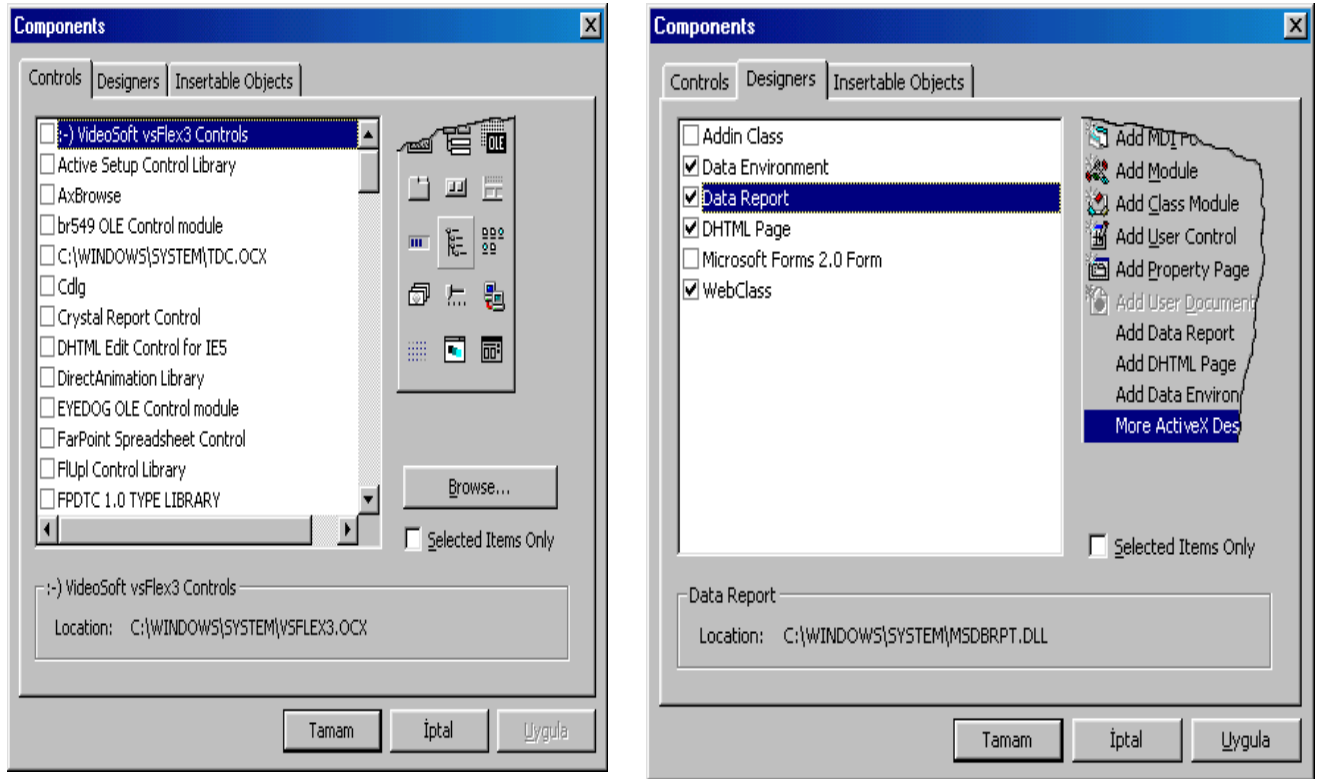
Ek Ders Ücreti Bordrosu																			
Report Header (ReportHeader)																			
Ek Ders Ücreti Bordrosu																			
Page Header (PageHeader)																			
Group Header (raporlar_Header)																			
FakülteAdı: FakülteAdi [raporlar]										%d									
Bölüm Adı: BolumAdi [raporlar]										Ait Olduğu Ay: AitOlduğuA									
Ücret Turu: ÜcretTuru																			

Şekil 11.3.2. Data Report Designer ile oluşturulmuş Örnek Rapor Ekranı-2

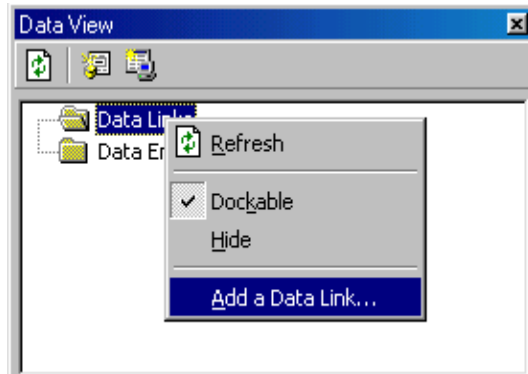
## 11.4 DATA REPORT DESIGNER İLE RAPOR OLUŞTURULMASI

Project menüsünden Add Data Environment seçilir. Bu seçenekle veritabanı tasarımcısı projeye eklenir. Eğer Project menüsünde Add Data Environment yer almıyorsa o zaman önce bunun Project menüsüne eklenmesi gerekmektedir. Bunun için aynı menüden Components ve ekrana gelecek pencereden de designer bölümüne geçiş yapılır.

Bu ekranda Designers listesinde yer alan Data Environment ve Data Report seçenekleri seçilir ve Tamam düğmesiyle çalışma ortamına geri dönülür. Bu işlemle data report seçeneği proje için eklenmiş olur.



Şekil 11.4.1. Data Report ve Data Environment Ekleme Ekranı



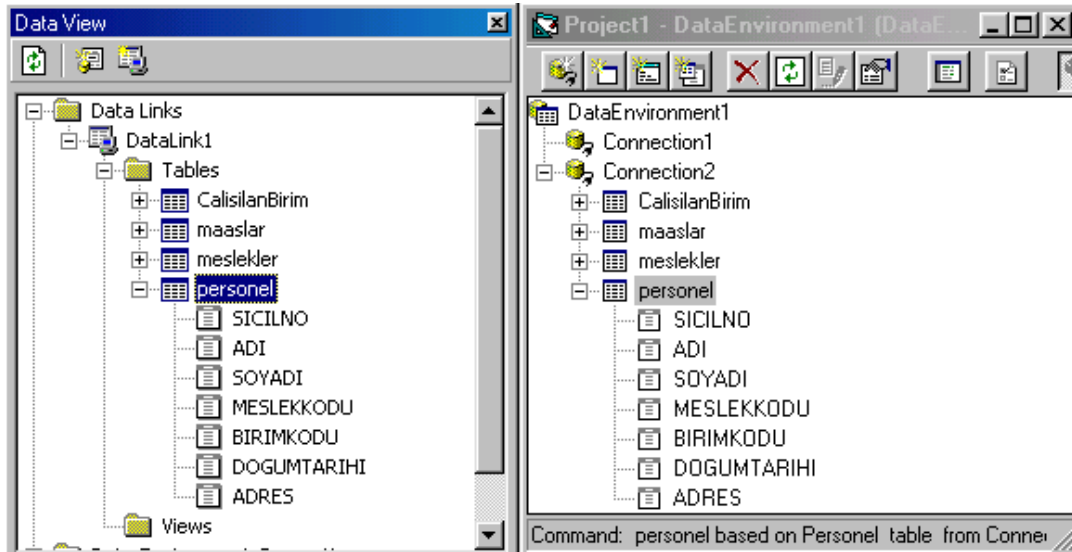
Şekil 11.4.2. Data View Ekranı

Ana menüden view - data view window seçeneği seçilir ve yan taraftaki görüntü ekrana gelir. Burada data links seçeneği üzerinde iken sağ mouse seçeneği ile "Add a Data Links" seçilir. Ekrana gelen Data Link Properties (Veri bağlantısı özellikleri) ekranından form hazırlamada olduğu gibi istenilen veri tabanına bağlantısı sağlanır. Daha sonra da rapor oluşturulur.

**Data Link Properties** (Veri bağlantısı özellikleri) diyalog kutusundan önce **Microsoft Jet 3.51 Ole DB Provider** seçilir ve Next (ileri) seçiminden sonra Connection alanına geçilir. Daha önce form hazırlarken yapıldığı gibi buradan database adı yazılır veya Browse ile bulunarak kullanılacak veritabanı seçilir. Test connection (bağlantıyı sına) ile bağlantı sınanır ve başarılı ise tamam butonuna basılarak veritabanına ait olan tabloların Data View ekranına gelmesi sağlanır.

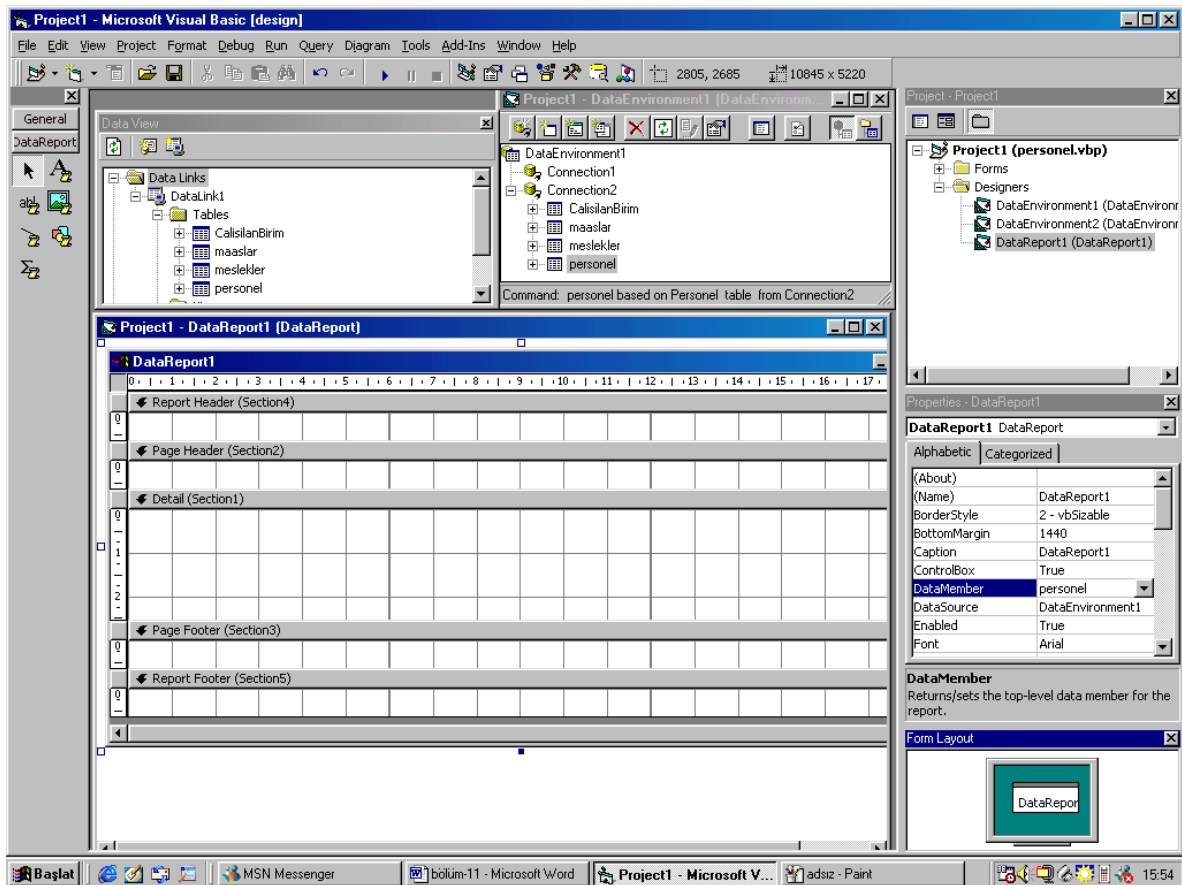
Data View penceresine gelen veritabanına bağlı olan tablolardan hangisi ile çalışılacaksa o tablo sürüklenip DataEnvironment penceresine bırakılır. Buraya kadar anlatılanlar veri tabanına bağlantının sağlanması için gerekli olan adımlardır.





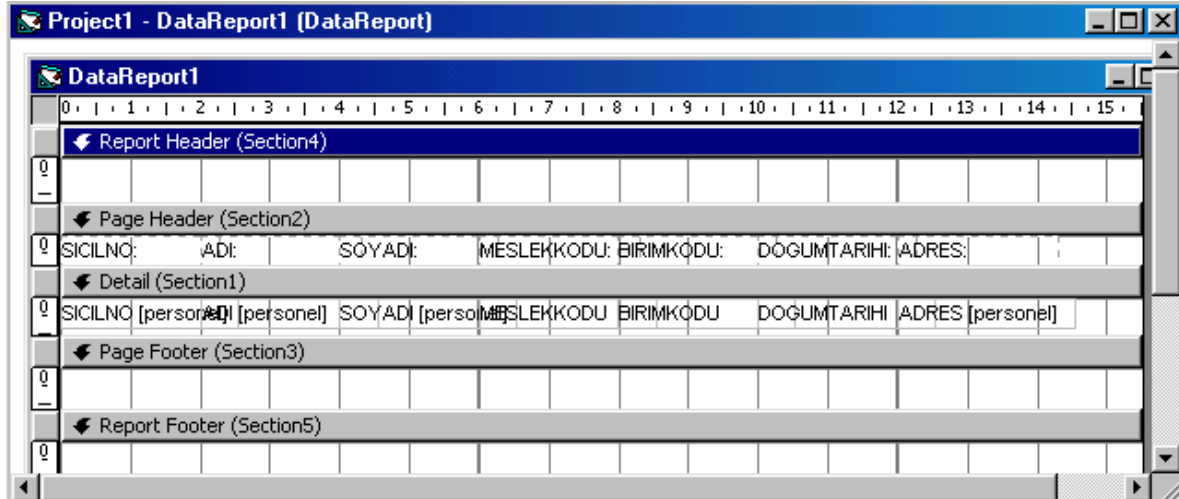
Şekil 11.4.3. Data View ve Data Environment Ekranı

Yeni rapor oluşturabilmek için **Project** menüsünden **Add Data Report** seçeneği tıklanır ve ekrana aşağıdaki görüntü gelir.

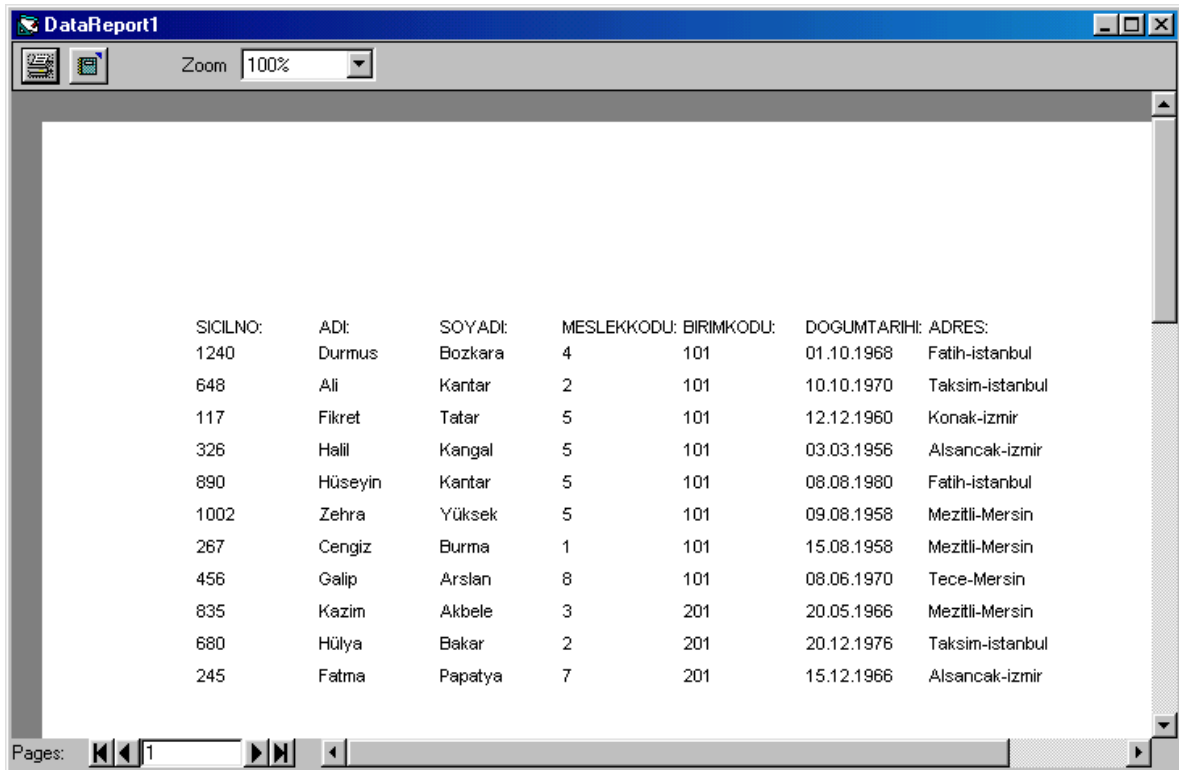


Şekil 11.4.4. Visual Basic Rapor Hazırlama Ekranı-1

Raporda veritabanı dosyasında yer alan tabloların içeriklerini göstermek için Data Report ekranı seçili iken properties ekranında **DataSource** özelliği DataEnvironment1 ve **DataMember** özelliği ise personel olarak ayarlanır. Daha sonra DataEnvironment penceresindeki tabloya ait veri alanları tek tek fareyle sürüklenerek rapor üzerine bırakılır.



Şekil 11.4.5. Visual Basic Rapor Hazırlama Ekranı-2



Şekil 11.4.4. Visual Basic ile Hazırlanmış Rapor Ekranı

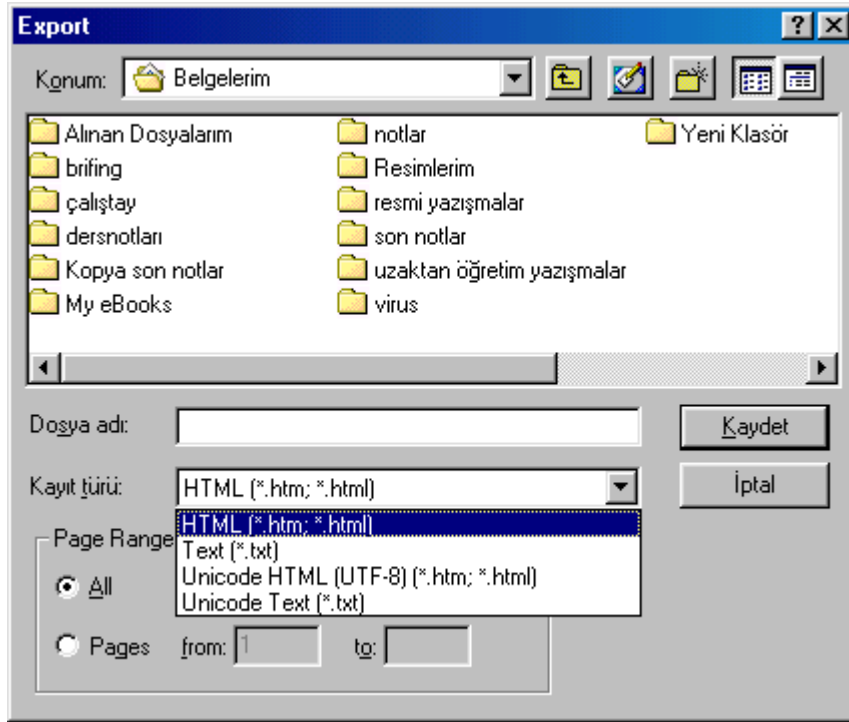
Rapor hazırlanmış ve çalıştırılarak sonuç elde edilmiştir. Raporu yer alan zoom kutusuyla rapor belli bir % lik oranda büyük yada küçük gösterilebilir. Rapor sayfasına dikkatle bakıldığında sol üst köşede iki adet buton alanının olduğu görülür. Bunlar ve kullanım amaçları aşağıdaki gibidir.



Raporu yazdırmak için kullanılır. Windows yazdır ekranından farklı olmayıp kullanımı aynıdır.

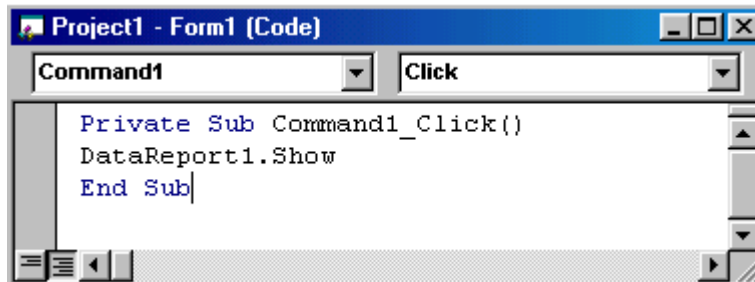


Rapor verileri html veya txt uzantılı olarak bir dosyaya dönüştürülerek kaydedilebilir. Buna ait görüntü ve dosyaya kaydedilebilecek formatlar aşağıdaki gibidir.



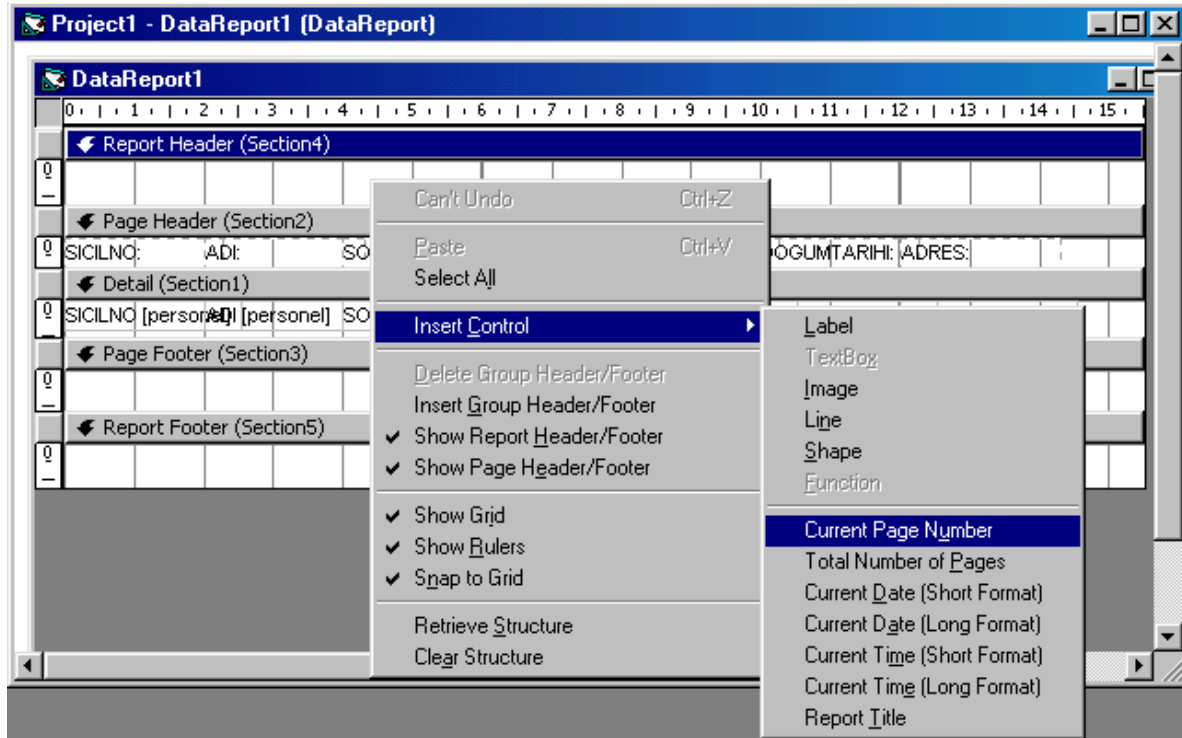
Şekil 11.4.5. Rapor Dönüştürme Ekranı

Rapor üzerine ilgili veri alanları bırakıldıktan sonra projeye ait olan form1 üzerine bir command butonunun yerleştirilmesi ve Click olayına aşağıdaki kodun yazılması gerekir. Daha sonra ilgili butona basıldığında rapor elde edilir.



Şekil 11.4.6. Visual Basic Rapor Hazırlama Ekranı-3

Hazırlanmış olan rapora sayfa numarası, tarih, alt toplamlar, ortalama gibi bileşenler eklenebilir. Bunun için rapor tasarım sayfasında sağ Mouse ile tıklanarak açılan menüden Insert Control seçeneği ile istenilen bileşenler eklenebilir. Buna ait ekran görüntüsü aşağıdaki gibidir.



Şekil 11.4.7. Rapora Bileşenler Ekleme Ekranı

## KAYNAKLAR

- Yaşar Gözüdereli, Veritabanı Programlamaya Giriş, Byte Dergisi Eğitim Dizisi, Temmuz 2003
- Yaşar Gözüdereli, Veritabanı Programlama II, Byte Dergisi Eğitim Dizisi, Şubat 2004
- Osman Nihat Şen, Oracle, SQL, SQL\*Plus, PL/SQL ve Veritabanı Yönetimi, İkinci Baskı, Beta Yayınları, İstanbul, 2000
- İhsan Karagülle, Zeydin Pala, Visual Basic 6.0 Pro, Türkmen Kitabevi, ISBN : 975-6812-08-7, İstanbul, 1999
- Aydın Şekihanov, Oracle8i A Practical Guide To SQL, PL/SQL, Developer 6, Atılım University Publications, 2001
- Oracle 8i Enterprise Edition Documentation
- Bilkent Üniversitesi Bilgisayar Merkezi, Veri Tabanı Yönetim Sistemine Giriş ve SQL, Temmuz 1996
- [http://iletisim.marmara.edu.tr/bilisim/veri%20modelleri\(csutcu%20dr%20tezinin%20bir%20bolumu\).pdf](http://iletisim.marmara.edu.tr/bilisim/veri%20modelleri(csutcu%20dr%20tezinin%20bir%20bolumu).pdf)
- <http://www.yazilimgrubu.com/dokuman.php?sayfa=2&no=%2010>
- <http://www.yazilimgrubu.com/dokuman.php?sayfa=2&no=%2011>
- <http://www.yazilimgrubu.com/dokuman.php?sayfa=2&no=%2012>
- <http://www.farukcubukcu.com/downloads/sqlkomutlari.doc>
- <http://www.farukcubukcu.com/downloads/sqlkomutlari.doc>
- [http://www.bto.yildiz.edu.tr/dosyalar/ebt2\\_konu5.doc](http://www.bto.yildiz.edu.tr/dosyalar/ebt2_konu5.doc)
- [http://www.bto.yildiz.edu.tr/dosyalar/ebt2\\_konu4.doc](http://www.bto.yildiz.edu.tr/dosyalar/ebt2_konu4.doc)
- <http://www.verivizyon.com>
- <http://www.verivizyon.com/detail.asp?cid=110>
- <http://www.verivizyon.com/detail.asp?cid=96>
- <http://cism.odtu.edu.tr/2002-7/veritabanı.php>
- <http://serdar.ktg.com.tr/oracle.htm>
- [http://www.geocities.com/cakmak\\_cengiz/](http://www.geocities.com/cakmak_cengiz/)
- [http://www.olgun.com/main\\_sql\\_destek.asp](http://www.olgun.com/main_sql_destek.asp)
- <http://bornova.ege.edu.tr/~sengonca/bolum1.html>
- <http://bornova.ege.edu.tr/~sengonca/bolum9.html>
- <http://www.beyazgroup.com/guvenlik.htm>
- <http://ted.see.plymouth.ac.uk/andyhandouts/ISAD324/Getting%20Started%20With%20Oracle%20Forms%20Builder.doc>
- <http://kocbey43.sitemynet.com/ora1.html>
- <http://kocbey43.sitemynet.com/ora2.html>
- <http://kocbey43.sitemynet.com/ora3.html>
- <http://kocbey43.sitemynet.com/ora4.html>
- [http://www.godoro.com/Divisions/Ehil/Mecmua/Magazines/Articles/txt/html/article\\_FormBuilders.html](http://www.godoro.com/Divisions/Ehil/Mecmua/Magazines/Articles/txt/html/article_FormBuilders.html)
- [www.oracle.com](http://www.oracle.com)