T.C.
FIRAT ÜNİVERSİTESİ
MÜHEDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

GÖRSEL BİR DİL KULLANILARAK WEB ÜZERİNDEN VERİ TABANLARINA ERİŞİM VE SANAL KÜTÜPHANE UYGULAMASI

Özgür BAĞKAN

BİTİRME ÖDEVİ

DERS SORUMLUSU : Ast. Prof. Dr. HASAN H. BALIK

ELAZIĞ
2001

T.C.
FIRAT ÜNİVERSİTESİ
MÜHEDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

GÖRSEL BİR DİL KULLANILARAK WEB ÜZERİNDEN VERİ TABANLARINA ERİŞİM VE SANAL KÜTÜPHANE UYGULAMASI

Özgür BAĞKAN

bitirme ÖDEVİ

Bu bitirme ödevi, .../...../..... tarihinde, aşağıda belirtilen jüri tarafından Oybirliği /Oyçokluğu İle Başarılı / Başarısız olarak değerlendirilmiştir.

(İmza) **(İmza)** **(İmza)** **(İmza)** **(İmza)**

Danışman

Ast. Prof. Dr. Hasan H. BALIK

Bu bitirme ödevi ...../...../..... tarih ve ..... sayılı kararıyla onaylanmıştır.

ÖNSÖZ

Bu çalışmada web üzerinden sql veritabanlarına erişimin delphi programlama dili kullanılarak nasıl yapılacağı anlatılmıştır.Bu proje bir sanal kütüphane uygulamasıdır. Web sayfalarına dinanizm katmak için isapi uygulaması kullanılmıştır.

Çalışmada projenin en iyi şekilde anlaşılması için gerekli tüm konular ayrıntılı bir şekilde açıklanmaya çalışılmıştır.İlk olarak sql veritabanı sorgulama dili ile ilgili kavramlar yeterli derecede anlatılmıştır.Daha sonra sql ve delphi programlama dili arasındaki bağlantılar, delphinin veritabanı özelikleri anlatılmıştır.İsapi ve İsapi uygulamaları anlatılmış ve projeden örnekler verilmiştir.Son olarakta sanal kütüphane projesinin bilgisayar yazılım mühendisliği gereklerine göre nasıl inşa edildiği anlatılmıştır.

TEŞEKKÜR

Sanal kütüphane uygulamasının gerçekleştirilmesini anlatan bu bitirme ödevini hazırlamamda bana yardımcı olan Ast.Prof. Dr. Hasan.H.Balık'a, bizim için emek harcamış olan diğer tüm hocalarımıza , projeyi gerçekleştirme aşamasında bilgilerini benden esirgemeyen araştırma görevlisi Koray Gündoğan'a teşekkür etmeyi bir borç bilirim.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

GİRİŞ

Günümüzde statik web sayfalarının yerine dinamik web sayfalarının kullanımında hızlı bir artış gözlenmektedir.Bu teknoloji sayesinde kurumsal veritabanlarına ,hem kurum personelinin erişiminde kolaylık sağlanmakta hem de dış kullanıcılar için erişim olanağı yaratılmaktadır.İnternet kullanımı hızla yaygınlaştığı için internet üzerinden veritabanlarına erişim başta elektronik ticaret olmak üzere çok sayıda yeni uygulamanın gelişmesine yol açmıştır.Bu uygulamalar arasında internet üzerinde sorgulama ,sipariş etme ,rezervasyon, kayıt yaptırma gibi birçok uygulama sayılabilir.İşte bu kitabın konusu olan sanal kütüphane projesi de web üzerinden veritabanlarına erişimi gösteren güzel bir örnek olmaktadır.

1. SQL'e GENEL BİR BAKIŞ

SQL (**S**tructured **Q**uery **L**anguage) veri tabanlarındaki verileri işlemek için kullanılan yapısal sorgulama dilidir.Veritabanını, verilerin depolandığı tablolar şeklinde düşünebiliriz. Tablolar ise ad, soyad, telefon gibi alanlardan (field) oluşur. Bu alanların tamamı ise bir kayıtı (record) oluşturur.Veritabanı tek tablodan oluşabileceği gibi birden fazla tablodan da oluşabilir. Yüzlerce tabloya sahip veritabanlarınız olabilir. Yine bu tablolarda 100 kayıt tutabileceğiniz gibi milyonlarca kayıtta tutabilirsiniz.

İndeksler(index) veritabanlarını hızlandırmak için kullanılır. Bir tabloda bir veya birden fazla indeks olabilir. İndeksleri seçerken en çok işlem yaptığınız alanları seçmeniz iyi olacaktır. Mesela raporlarınızı daha çok tarihe göre ve il bazında alıyorsanız tarih ve il_kodu alanlarını index tanımlamalısınız.
Tablolarda alan(field) ve kayıt(record) karıştırmamanız lazım. Mesela ad, soyad, telefon ve adres'ten oluşan bir tablonuz var. Bunu bir excel sayfası olarak düşünebilirsiniz.

Burada yukarıdan aşağı doğru olan kolonlar alan'dır. Yani ad alanı, soyad alanı gibi. Soldan sağa doğru olan sütunlar ise kayıttır. Yani (Ali, Akın, 0 216 123 45 67, ...) bir kayıttır.

Veritabanları etkin kayıtın yerini göstermek için kürsor(cursor) denen bir yapı kullanırlar. Mesela siz programla Zafer Yıldırım isimli kayıtı görüntülediğiniz zaman veritabanında kürsor bu kayıtı gösterir.

Birde dataset kavramı var. Dataseti şöyle açıklayabiliriz: Birden çok tablodan elde ettiğimiz veri topluluğu. Mesela müşterilerinizin adını ve adresini sakladığınız bir tablo, siparişlerini sakladığınız bir tablo ve siparişlerin detayını sakladığınız bir tablo olmak üzere 3 tablonuz var. Siz Ali Taş isimli müşterinin Ocak ayında yaptığı siparişlerin ayrıntılarını görmek istiyorsunuz. Bu 3 tablodanda veri alırsınız. Bu bir dataset olur.

Tablo 1 Örnek Tablo Gösterimi

Ad	Soyad	Telefon	Adres
Ali	Akın	0 216 123 45 67	....
Zafer	Yıldırım	0 212 123 45 67	...

Veritabanlarını iki kısımda ele alabiliriz. Lokal veritabanları ve C/S veritabanları.Lokal veritabanları tek bir makinede çalışmak üzere tasarlanmış veritabanlarıdır. Bu verilere sadece sizin programınız erişecek ve başkaları bu verilere erişmeyecekse lokal veritabanını kullanabilirsiniz. Paradox, dBase ve Access lokal veritabanlarıdır.

C/S Veritabanları: Veritabanınız bir server üzerinde durur ve birden fazla kullanıcı (client) bu verilere erişir ve işlem yapar. C/S veritabanları bir kayıta birden fazla kişinin aynı anda erişmesi gibi olayları kendileri kontrol ederler ve bu tip durumların üstesinden nasıl geleceklerini bilirler. Interbase, Oracle, SQL Server, Sybase, Informix, DB2 kullanılan C/S veritabanlarıdır.

C/S veritabanlarının bir dezavantajı veritabanını kullanan kullanıcı başına para ödemenizdir. Şirketler burada iki farklı lisans yöntemi sunuyorlar. Birincisi kullanan kullanıcı başına para ödüyorsunuz (per seat). 40 tane kullanıcınız(client) varsa 40 client lisansı almalısınız. Diyelimki kullanıcı sayınız arttı ve 50 oldu. 10 tane daha client lisansı almanız gerekir. İkinci bir yöntem ise veritabanına aynı anda kaç kullanıcı bağlanıyorsa o kadar lisans almaktır (per connection). Mesela 100 kullanıcınız veritabanını kullanıyor ancak aynı anda en fazla 30 kişi veritabanına bağlanıyorsa 30 lisans almalısınız.

C/S veritabanları içinde şu anda Oracle en iyisi. Onun dışında Türkiye piyasasında SQL Server'da çoğunlukla kullanılmakta. Benim tercihim Oracle'dan yana. Eğer C/S veritabanlarında kendinizi geliştirmek istiyorsanız Oracle veya SQL Server seçin.

Lokal veritabanları ile da birden çok kullanıcının kullandığı programlar yazmak mümkün. Ancak çoğu şeyi programla halletmeniz gerekiyor. Genellikle de problemler çıkmakta.

1.1. Tek Katmanlı, İki Katmanlı ve Çok Katmanlı Database Yapıları

Tek katmanlı(single-tier): Genel olarak lokal veritabanları tek katmanlıdır. Tek katmanlı yapıda program veritabanına direk ulaşır ve yapılan işlemler (kayıt ekleme, kayıt silme, değiştirme) anında gerçekleştirilir.

İki katmanlı(two-tier): Burada program client tarafında çalışır. Client tarafında çalışan program gerekli sürücüleri kullanarak serverdaki veritabanına ulaşır.

Çok katmanlı(multi-tier): Burada program yine client tarafında çalışır. Program database server'la direk bağlantı kurmaz. Server tarafında çalışan bir application server ile bağlantı kurar. Bu yapı genellikle güvenlik ve hız amacıyla kullanılır.

1.2. BDE (Borland Database Engine)

BDE Delphi'nin lokal ve C/S veritabanlarına bağlanmak için kullandığı dll ve uygulamalardır. C/S veritabanlarına bağlanmak için Delphi'nin C/S sürümünü kullanmak zorundasınız. Bu sürümle birlikte gelen SQL Links, BDE tarafından C/S veritabanlarına bağlantı için kullanılır.

1.2.1 Delphi BDE Gibi Bir Yapıyı Niye Kullanıyor

Normalde veritabanlarının yapıları ve API'leri farklı farklıdır. BDE programcıyı tüm bu yapıları öğrenmekten kurtararak daha üst düzey komutlarla program yapımına imkan tanır.

Delphi ile gelen sürücüler kullandığınız Delphi sürümüne göre değişir. Delphinin tüm sürümleri ile Paradox ve dBase bağlanmayı sağlayan sürücüler gelir. Bu

sürücüler STANDART olarak adlandırılır ve paradox ve dBase ile yapacağınız tüm işlemleri yapmanızı sağlar. Delphi'nin C/S sürümü ile Oracle, SQL Server, Interbase, Sybase, Informix gibi C/S veritabanlarına bağlanmanızı sağlayacak sürücülerde gelir.

1.2.1.1.Alias (Takma isim/Rumuz)

Alias'lar veritabanlarına bağlantı ve bağlantının özelliklerini ayarlamak için kullanılır. Bir alias BDE'ye hangi tür bir veritabanına bağlanacağı, veritabanı dosyalarının diskte nerede olduğu gibi bilgileri bildirir. Ayrıca eğer C/S bir veritabanı kullanıyor iseniz açılış modu, kullanıcı ismi, BLOB alanların büyüklüğü gibi verileri BDE'ye bildirir.

1.2.1.2.Lokal Veritabanları İçin Alias Oluşturma

1.BDE Administrator programını çalıştırın.
2. Object menüsü altından New... komutunu verin veya klavyeden Ctrl+N tuşlarına basın.
3.Database Driver Name kısmından STANDART'ı seçin.
4. Alias'ın ismini yazın ve Object menüsünden Apply komutunu verin.
5. Definition kısmında Default Driver kısmından kullandığınız veritabanını seçin.
6. Path kısmına veritabanınızın olduğu yeri seçin. 7. Object menüsünden Apply komutunu verin veya klavyeden Ctrl+A tuşlarına basın.

Bu dil yardımıyla veritabanlarındaki tüm işlemler yapılabilir. Backup almadan tutunda bir tabloya veri girmeye varıncaya kadar herşey.

SQL'i şu anda piyasada bulunan hemen hemen her veritabanında kullanabilirsiniz. SQL'de her veritabanında kullanılan ortak ifadeler olmasına karşın, veritabanlarının kendine özgü ifadeleri de vardır. Mesela Oracle'da SQL ile yapabildiğiniz bazı şeyleri başka veritabanlarında yapamayabilirsiniz.Şimdi sql dili ile ilgili bazı temel kavramlara kısaca bakalım.

1.3. Genel Kavramlar

Table : Database'de saklanan kolonların birleşiminden oluşan kümedir. Table'ın data tipi yoktur.
Row (sıra): Tek kayıt demektir.
Column : Table'daki kolon adına ait kayıtlardır. Örneğin, 'Ad' kolonu demek 'Ad' kolonuna girilen verilerin tümüdür.
Field : Kolon başlıkları ile kayıt başlığı olanlardır.
Primary Key : Unique + not null (Tek olmalı ve boş geçilemez.)
Foreign Key : Primary key gibidir. Fakat null değerler alabilir.

Table'lar ilişkisel veri tabanı(compact) olmalıdır. Tablolar arasında bir ilişki kurulmalıdır.

Database'e üzerinde giriş,değişiklik,silme vb. İşlemler sql ile olur. Tablolar fiziksel olarak gözükmezler. Database içinde saklıdırlar.

Data Dictionary: Database'deki kullanıcılar, yaratılan tablolar vb. Nesneler hakkında detaylı bilgiler bulunan tablodur.

Belli başlı fonksiyonlarla ulaşılabilir. Bunlardan bazıları şunlardır.İnsert, update,delete ile var olan bir tablo üzerinde değişiklikler yapılabilir. Create,alter,drop,rename ile yapısal değişiklikler yapılabilir.Commit,rollback ile yapılan işleri onaylar veya geri alır, iptal edilebilir.

SQL temel olarak şu ifadelerle kullanılır. SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING, UPDATE, DELETE, INSERT.

Burada kullandığımız SQL cümleleri ISCI ve PERSONEL adlı bir tablo üzerine yazılmıştır.

Select: Tablodan seçmek istediğimiz alanları belirtmek için kullanılır. Eğer tablodan tüm alanları seçmek istiyorsak o zaman alan isimleri yerine * işareti konur.
From: Üzerinde işlem yapılacak tablo/tabloları belirtmek için kullanılır.

Where: Tablodan eğer tüm kayıtları değilde istediğimiz bazı kayıtları elde etmek istiyorsak, örnekte maaşı 250 milyondan fazla olan işçilerin numarası ve adi gibi, o zaman buraya istediğimiz kriteri yazarız.

```
SELECT    ISCI_NO, ISCI_ADI
FROM ISCI
WHERE MAAS>250000000
```

Distinc: Birbirinin aynı olan satırların listelenmemesi için bu ifade kullanılır. Mesela ISCI tablosunda bulunan birbirinin aynı olmayan isimleri listelemek istersek

```
SELECT  DISTINCT ISCI_ADI
FROM ISCI
```

şeklinde bir SQL ifadesi yazarız.

In: Koşul belirtirken kullanırız. Mesela ismi AHMET, ALİ veya MUSTAFA olan işçilerin bilgilerini listelemek için

```
SELECT  * FROM ISCI
WHERE ISCI_ADI=’AHMET’ OR ISCI_ADI=’ALİ’ OR ISCI_ADI=’MUSTAFA’
```

şeklinde bir ifade kullanırız. Bunun yerine

```
SELECT  * FROM ISCI
WHERE ISCI_ADI IN (’AHMET’ ,’ALİ’ ,’MUSTAFA’)
```

ifadesini de kullanabiliriz. Yani listenin içindeki herhangi bir değerin bulunması kayıtın seçilmesi için yeterlidir.

Like: Eğer aradığımız kayıtın bulunması için tam bir karşılaştırma yapamıyorsak o zaman kullanırız. Mesela isminin baş harfi A ile başlayan isimleri bulmak için

SELECT * FROM ISCI
WHERE ISCI_ADI LIKE ‘A**%**’ ifadesi kullanılır.

% işareti uzunluğu önemsiz olmak üzere yazıldığı yere her türlü ifade gelebilir anlamındadır.

? işareti ise bir karakter olmak üzere her türlü değeri alabilir anlamındadır. Mesela isminin sondan üçüncü harfi A, ve son harfi Z olan kayıtları listelemek istersek sondan ikinci harfin ne olduğu önemli değildir. O zaman o harf yerine aşağıda görüldüğü üzere ? işaretini kullanırız.

SELECT * FROM ISCI
WHERE ISCI_ADI LIKE ‘**%A?Z**’ ifadesi kullanılır.

Between: Koşul belirtirken iki değer arasını belirtmek için kullanılır. Örnek: Yaşı 30 ile 40 arasındaki işçilerin kayıtlarını listelemek için

SELECT * FROM ISCI
WHERE YAS BETWEEN 30 AND 40

ifadesi kullanılır. Bunu aynı zamanda aşağıdaki ifade ile de yapabilirsiniz. BETWEEN yazım kolaylığı sağlar.

SELECT * FROM ISCI
WHERE YAS>=30 AND YAS<=40

Sum: Seçilen değerlerin toplamını bulur. İşçilerin aldığı toplam ücreti görmek için

SELECT SUM(UCRET)
FROM ISCI

ifadesi kullanılır.

Max, Min, Avg: Verilen değerin en büyüğünü, en küçüğünü ve ortalamasını bulur. 1999 yılında giren işçilerin en yüksek ücretinin, en düşük ücretinin ve ortalamasının ne kadar olduğunu öğrenmek istersek aşağıdaki ifadeyi kullanırız.

```
SELECT  MAX(UCRET), MIN(UCRET), AVG(UCRET)
FROM ISCI
WHERE GIRIS_TARIHI>'01.01.1999'
```

MAX en büyük değeri, MIN en küçük değeri, AVG ise seçilen değerlerin ortalmasını bulur.

Order By: Tablodan seçtiğimiz kayıtları sıralamak için kullanılır. Yukardaki örnekte isimleri alfabetik sıra ile görmek istersek

```
SELECT  DISTINCT ISCI_ADIFROM ISCI
Order by  ISCI_ADI
```

yazarız. Eğer sıralamayı tersine çevirmek istersek

```
SELECT  DISTINCT  ISCI_ADI
FROM  ISCI
Order by  ISCI_ADI desc
```

yazarız.

Group By: Genelde istatistik amaçlar için kullanılır. Mesela hangi tarihte kaç işçinin işe alındığını bulmak için

```
SELECT GIRIS_TARIHI,COUNT(*)
FROM ISCI
GROUP BY GIRIS_TARIHI
```

yazmanız yeterli olacaktır. Bu ifade size gün bazında kaç işçinin işe alındığını gösterecektir. Eğer belli bir tarihten önce ya da sonrasını isterseniz veya sadece sayının 10'dan büyük olduğu günleri görmek isterseniz o zaman ifadeyi şu şekilde yazmak gerekir

```
SELECT  GIRIS_TARIHI,COUNT(*)
FROM  ISCI
WHERE GIRIS_TARIHI>'01.01.1999'
GROUP BY GIRIS_TARIHI
HAVING COUNT(*)>10
```

HAVING, grup fonksiyonlarının kriterleri için kullanılır. SUM, COUNT vb. gibi.

Update: Tabloda bulunan bir istediğiniz bir veya daha fazla alanın güncellenmesi amacıyla kullanılır. Mesela işçilerin maaşlarına % 20 zam yapıldığını düşünürsek aşağıdaki ifade ile bunu tabloda gerçekleştirebiliriz.

```
UPDATE  ISCI
SET MAAS=MAAS*1.2
```

Eğer maaşlarla birlikte aldıkları primleri de %20 oranında artırmak isterseniz

```
UPDATE  ISCI
SET MAAS=MAAS*1.2 , PRIM=PRIM*1.2
```

şeklinde bir ifade kullanılır. Aynı zamanda WHERE ifadesini kullanarak sadece belli kayıtlar üzerinde güncelleme yapabilirsiniz.

Delete: Tabloda bulunan kayıtları silmek için kullanılır. Eğer

```
DELETE FROM ISCI
```

derseniz tüm kayıtları gönderirsiniz. DELETE ifadesini kullanırken dikkatli olun. Buradada yine WHERE ifadesini kullanarak sadece belli kritere uyan kayıtların

silinmesini sağlayabilirsiniz. Kötü bir örnek ama olsun, patron 45 yaşından büyük işçileri işten attı (burası Türkiye, olmaz demeyin) ve kayıtlarının silinmesi isteniyor. O zaman

```
DELETE  FROM  ISCI
WHERE YAS>45
```

ifadesi kullanılır.

Insert: Tablolara veri girişi yapmak amacıyla kullanılır.

```
INSERT INTO ISCI (ISCI_NO,ADI,SOYADI) VALUES (1000,'ALİ','BORAN');
```

Eğer giriş yaparken tablonun bütün alanları kullanılacaksa alan isimlerini vermeye gerek yoktur.

Not: UPDATE, DELETE ve INSERT ifadelerini kullanırken dikkatli olmalısınız. Eğer SQL tabanlı bir veri tabanı kullanıyorsanız bu ifadeleri veritabanlarının kendi tool'ları üzerinde kullanın. Çünkü Delphi ile gelen SQL Explorer'da işaretine basmayı unutursanız yaptığınız işlemin geri dönüşü olmayabilir. Yani en son yaptığınız işlemi Rollback yapamazsınız ve eğer gerçek veritabanı üzerinde yaptıysanız işlemi başınız bayağı ağrıyabilir veya o iş yerinde yazdığınız son SQL ifadesi olabilir. :-))

1.4. Tablolar Üzerinde İşlemler

İki Tablodan Birden Kayıt Seçmek; İşçilerin kimlik bilgilerinin ISCI_KIMLIK tablosunda tutulduğunu kabul ederek bizden ÇORUM doğumlu olanların listesinin istendiğini varsayalım. Tablolar birbirine ISCI_NO alanı üzerinden ilişkili olsun

.

```
SELECT  A.ISCI_NO, A.ISCI_ADI, B.DOGUM_YERI
FROM  ISCI A, ISCI_KIMLIK B
WHERE  B.DOGUM_YERI='ÇORUM'
AND  A.ISCI_NO=B.ISCI_NO
```

şeklinde bir ifade yazarak listemizi elde edebiliriz. Burada WHERE koşuluna yazdığınız sıranın pek bir önemi yoktur. Her şartta aynı sonuçları elde ederseniz. Fakat performans açısından biraz farkeder. Yukardaki ifade

```
SELECT  A.ISCI_NO, A.ISCI_ADI, B.DOGUM_YERI
FROM  ISCI A, ISCI_KIMLIK B
WHERE  A.ISCI_NO=B.ISCI_NO
B.DOGUM_YERI='ÇORUM'
```

ifadesinden daha hızlı çalışır. Çünkü ilk ifadede önce doğum yeri ÇORUM olan kayıtlar seçilir daha bu kayıtlara işçi tablosu birleştirilir. Sonraki ifadede ise önce tüm kayıtlar birleştirilir, bunların arasından doğum yeri ÇORUM olanlar seçilir.

1.4.1 Tabloların Yaratılması

Database'de verilerin saklanması amacıyla tablolar yaratılır. Tablo yaratabilmek için o kullanıcının buna yetkisi olmalıdır. Aynı zamanda limitsiz tablespace hakkına sahip olmalıdır.

Kullanım Şekli;

CREATE TABLE [tablo adı.]

Veritipleri

Varchar2(boyut): Boyut ile belirtilen max miktar kadar karakterdir.Max değeri 2000'dir.

Char(boyut) : Boyut ile belirtilen max miktar kadar karakterdir.Max. değeri 255'dir.

Number : e^{38} 'e kadar olan tüm sayısal değerlerdir.

Number(m,n) : m kadar (max $e^{38)}$ sayının n kadar ondalık alan için değer alır.

Date : Tarih ve saat değerlerini bir tutar.

Boolean : Mantıksal ifadeleri saklar. Yani doğru ise True, yanlış ise false.

Long : Max 2GB 'a kadar büyük olan alanlar için yer tutar.

Raw : Grafiksel yapıdaki veriler için tanımlanır.

Commit : Bütün yapılan işlemleri kesin olarak kalıcı olmasını sağlar. Böylece yapılan değişiklikleri varsa diğer kullanıcılarda görür.

Rollback x : Bütün yapılan işlemleri kesin olarak iptal eder.ROLLBACK dön teklindeki bir işlemle personel kaydının silinmesi ve yeni kayıt eklenmesi işlemi iptal edildi. Ama ilk yapılan değişiklik kaldı.Read uncommitted Commit olmadan görünmez. O halde böyle level yoktur.Read only Transaction işleminde sadece okuma var demektir.

Alter table : Yeni bir kolon eklemek, kolonun tipini veya uzunluğunu değiştirmek vb. yapısal değişiklikler yapılması için kullanılır. Eğer kolon üzerinde değişiklikler yapılacaksa dikkat edilmesi gereken koşullar vardır. Örneğin kayıt uzunluğu 15 iken uzunluğunu 10'a indirirsek kayıt içindeki bilgiler kesilir. Kolonlar ekleyebilir ve yapısal değişiklikler yapabiliriz. Constraint yapısını ekler, silebilir, enable ve disable yapabiliriz.

Kullanım Şekli;

ALTER TABLE [tablo adı]

Drop Table : Tabloyu fiziksel olarak siler. Rollback komutu ile silinen tablo geri getirilemez. Tabloyu ancak yetkisi olan kullanıcı silebilir.

Kullanım Şekli;

DROP TABLE [tablo adı]

Örnek: Personel tablosunu silelim;

DROP TABLE Personel

Rename ..**To**.. : Objelerin ismini değiştirmek için kullanılır. Otomatik olarak commit olur. Tabloyu ancak yetkisi olan kullanıcı silebilir.

Kullanım Şekli;

RENAME eski_isim TO yeni_isim

Örnek: Personel tablosunun adını pers olarak değiştirelim;

RENAME personel TO perso

Truncate Table : Tablodaki tüm kayıtları siler. Delete komutu gibi olmasına karşın o komuttan çok daha hızlı silme işlemi yapar. Rollback komutu ile silinen kayıtlar geri getirilemez. Otomatik olarak commit olur. Tabloyu ancak yetkisi olan kullanıcı silebilir.

Kullanım Şekli;

TRUNCATE TABLE [tablo adı]

Delete komutu ile Truncate arasındaki en önemli fark; Delete komutu kayıtları silmek için kayıtlarda boşluk bırakır. Truncate ise tamamen kayıtları temizler, yani başa sarar.

Örnek: Personel tablosundaki tüm kayıtları silelim;

TRUNCATE TABLE Personel

View: Bir tablo üzerinde sorgulama yapılması için kullanılan nesnedir. Fiziksel olarak herhangi bir yerde saklanmaz. Avantajları;

1.Database erişimini kısıtlar. Böylece sadece sorgulanan verileri gözükür.

2.Sorgulamaları kolaylaştırabilir.

3.Datayı bağımsız olarak gösterebiliriz.

Kullanım Şekli;

CREATE VIEW view_adi [alias]

AS ...

View_adi : Yaratılan view sorgu adıdır..

Alias : Yaratılan işlemin adıdır.

WITH CHECK OPTION [CONSTRAINT View objesine hatalı işleme yapılmasını engeller.

Örnek: Departman numarası 12 olan peroneller için perview isminde bir view olşturalım;

```
CREATE VIEW persview
AS SELECT *
FROM personel
WHERE dept_id=12
```

Drop View : Yaratılan view objesini siler.

Kullanım Şekli;

DROP VIEW view_adı

1.5. İndex Yaratma

Indexler, bir tablonun istenilen kolonlarına daha hızlı erişim olanağı sağlamak için kullanılır. Tablodaki kayıtlar üzerinde giriş/çıkış işlemleri yapılırken dataya daha hızlı ulaşılması sağlanır. Primary key tanımlanan kolonlar için otomatik olarak index yaratılır. Index en fazla 16 kolondan oluşur. Bir kolon tipi long veya long raw olamaz.

Kullanım Şekli;

CREATE INDEX index_adi
ON tablo_adı(kolonlar)

Örnek: Personelin adına ve soyadına göre index oluşturalım;

CREATE INDEX personel_inx
ON personel(ad,soyad);

Böylece ad kolonu öncelikli olmak kaydıyla birlikte soyadına göre sıralama yapar, yani index oluşturur. Önce ada göre sıralar, eğer aynı isimden birden fazla kayıt oluşursa bu sefer soyad kolonundaki değerlere göre sıralama oluşturur.

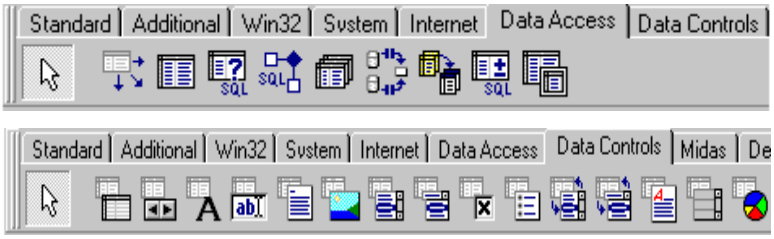
2. DELPHİ İLE VERİTABANI PROGRAMLAMAYA GENEL BİR BAKIŞ

Delphi'nin tercih edilmesinin en büyük nedenlerinden biriside iyi bir veritabanı desteğine sahip olması ve veritabanı programı yazmayı son derece kolaylaştırmasıdır.Delphi ile hiç kod yazmadan bir veritabanı programı yapabilirsiniz.Burada herşeyin anlatılması mümkün değil. sadece veritabanının genel olarak yapısını ve veritabanı programcılığının temelini anlatmaya çalışılacağım.

Bileşen paletinde **Data Access** ve **Data Controls** sayfalarında bulunurlar. Genel olarak bir veri tabanına bağlanıp veriler üzerinde insert, update,delete veya belli kayıtların görüntülenmesi için kullanılır. Bileşenler şekil 2.1 de görülmektedir.

2.1. Data Access

Data Access sayfasındaki bileşenler unvisible (yani program çalıştığı zaman ekranda gözükmeyen) bileşenlerdir. Bu bileşenler **Data Controls** sayfasındaki bileşenler yardımı ile görüntülenecek veriler için veritabanları ile köprü vazifesi görürler.



Şekil 2.1 Data Access

2.2. TTable

En önemli iki özelliği Database Name'i ve Table Name'dir. Database Name'e BDE içinden tanımladığınız herhangi bir alias'ı, projeniz içindeki Database bileşeninin Database Name'ini (bunlar combobox içinde otomatik olarak gelirler) veya paradox vb. gibi tablolar için tabloların bulunduğu dizinin adını verebilirsiniz. Bu bileşenin diğer önemli özelliklerinden biri de Index Name-Index Fields özelliğidir. Bu özelliği herhangi bir kayıta ulaşmak için kullanılır. Herhangi bir kayıta ulaşmak veya istenilen bir kayıtı bulmak için genel olarak üç çeşit yöntem kullanılır. Bunlar

1-FindKey()
2-Locate()
3-Lookup()

prosedürleridir. Birbirinden farkları şudur. Eğer findkey'i kullanıyorsanız index name ve index fields özellikleri belirtilmiş olmalıdır. Bu fonksiyon istenilen kayıt bulunmuşsa True aksi halde False değerini döndürür. Kullanılışı şu şekildedir :
Table1.FindKey([değişken1,değişken2,…]) gibi. Buradaki değişken sayısı index fields özelliğinde tanımlanan veya index name ile belirtilen indexin sahip olduğu alan sayısına eşit olmalıdır ve o alanlara karşılık gelen değerler verilmelidir.

Örnek: Index Fields=Numara olsun. Bu durumda kod şu şekilde olmalıdır. Table1.FindKey([2500]) gibi. Burada indekste belirtilen alanın tipi ile koda yazdığımız tip birbirini tutmalıdır. Eğer Index Fields=Adi;Soyadi şeklinde ise kod Table1.FindKey([‘Ahmet’,’SAVAŞ’]) şeklinde olmalıdır.

Locate prosedürü de FindKey gibi çalışır. Fakat bunda alan isimlerini de kendiniz verirsiniz. Eğer belirlediğiniz alanlara ait bir index varsa kullanılır, yoksa sıralı arama yapılır. Kullanılışı:
Table1.Locate(‘adi;soyadi’,VarArrayof([‘Ahmet’,’SAVAŞ’]),[loCaseInsensitive,loPartialKey]) şeklindedir.
loCaseInsensitive: Büyük harf-küçük harf ayrımı yapılmaz.

loPartialKey: Bunu kullanırsanız eğer sadece SAVAŞ'ı değilde eğer SAVAŞÇI da varsa onuda bulabilirsiniz.

Lookup da aynı şekildedir. Farklı yanı ise şudur; FindKey ve Locate de cursor bulunan kayıtın üstüne gider, lookup da ise nerde iseniz orda durur. Bir de bu fanksiyonları kullanırken dikkat edeceğiniz diğer bir husus en son yaptığınız işlemdir. Bu prosedürlerden herhangi birini çağırdığınız zaman en son yapılan işlem otomatik olarak kaydedilir. Örneğin iptal etmeniz gereken bir işlem varsa veya tablo insert modundaysa bu durumlarıda kontrol etmelisiniz. Diğer sık kullanılan prosedürleri ise Edit, Insert, Post'dur.

Tablonun hangi durumda olduğunu Table1.State ile öğrenebilirsiniz. Örneğin Table1.State in [dsEdit] tablonun edit modunda olduğunu gösterir.

2.3. Tquery ve TUpdateSQL

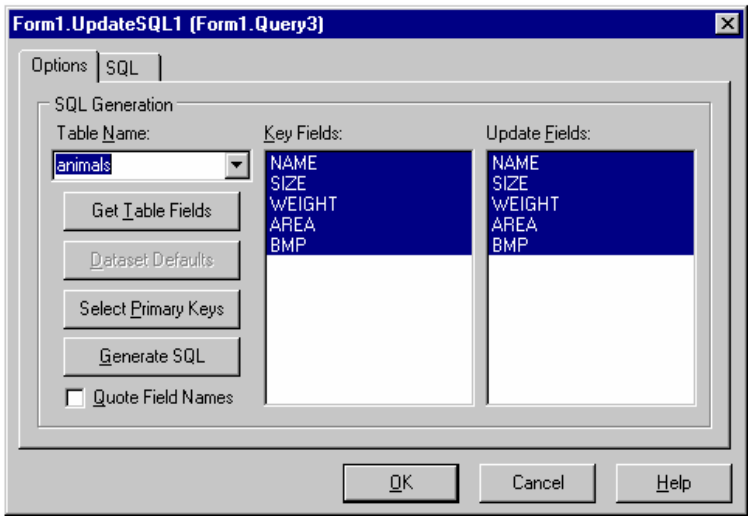
Bu bileşende TTable bileşeni ile hemen hemen aynıdır. Fakat bu bileşen ve SQL yardımı ile kayıtlar üzerinde sıralama, sadece belli kayıtları görüntüleme vb. işlemler çok daha rahat yapılabilmektedir. Aynı şeyler TTable bileşenin Filter, Range gibi özellikleri kullanılarak da yapılabilir. Fakat, performans açısından bakıldığında TQuery'leri kullanmak her zaman faydalıdır. Query'ler normalde **Read-Only**'dirler, yani kayıtlar üzerinde değişiklik yapamazsınız. Eğer kayıtlar üzerinde değişiklik yapmak istiyorsanız, **RequestLive** özelliğini True yapmalısınız. Bu şekilde kayıtlar üzerinde değişiklik yapıp, yapılan değişiklikleri table'da olduğu gibi kaydedebilirsiniz. Fakat SQL cümleciğiniz birkaç tablodan veri alıp getiriyorsa o zaman RequestLive özelliğini kullanmazsınız. Bu şekildeki query'ler üzerinde değişiklik yapabilmeniz için önce **CachedUpdates** özelliğini True yapmalısınız. (Bu arada RequestLive, False olmalıdır.) CachedUpdates özelliği True yapıldığında kayıtlar üzerinde güncelleme, değiştirme ve silme yapabildiğinizi göreceksiniz. Fakat bu değişiklikler sadece programda kalır ve fiziksel veritabanını etkilemez. Yaptığımız değişikliklerin kalıcı

olması olması için **TUpdateSQL** bileşenini kullanırız.Bu işlem için aşağıdaki adımları takip etmelisiniz.

1. TQuery ve TUpdateSQL bileşenlerini yerleştirin
2. TQuery'nin SQL ifadesini yazın.
3. TQuery'nin UpdateQbject özelliğini UpdateSQL1 olarak seçin(veya ne isim verdiyseniz)

UpdateSQL1 bileşeninin seçip mouse'un sağ tuşuna basın. Oradan **UpdateSQL Editor…**'ü tıklayın.Şekil 2.2 karşınıza gelecektir.

İlk önce üzerinde değişiklik yapmak istediğiniz tabloyu seçmelisiniz. Daha sonra Key Fields ile belirtilen genellikle sizin üzerinde değişiklik yapmadığınız alanlar seçilmelidir. Numara vb. Daha sonra değiştirilecek alanlar Update Fields da belirtilen alanlardan seçilmelidir. Daha sonra ise **Generate SQL** butonuna bastığınızda sizin için gerekli SQL'lerin oluştuğunu göreceksiniz. Daha sonra kod içinde istediğiniz herhangi bir yerde UpdateSQL1.ExecSQL() prosedürünü kullanarak yaptığımız işlemin kalıcı olmasını sağlayabiliriz.



Şekil 2.2 Updatesql

ExecSQL'in alacağı parametreler;

ukModify : Güncelleme işleminin kalıcı olması için
ukInsert : Yeni girilen bir kayıtın veritabanında olması için
ukDelete : Sildğiniz bir kayıtın veritabanından da silinmesi için. TTable üzerinde geçerli olan işlemlerin birçoğu TQuery içinde geçerlidir.

2.3.1.TQuery İle Parametre Kullanımı Ve Prepare Metodu

Burada TQuery ile parametre kullanımını ve bunun sağlayacağı avantajları inceleyeceğiz.

Öncelikle örnek tabloyu yaratalım. Tablonun adı müşteri olsun.
Müşteri

Kodu Autoinc
Adı string
TQuery ile bu tablodan tüm kayıtları istemek için bildiğiniz gibi:
"select * from müşteri "
demek yeterlidir. Peki isminin içerisinde "ve" kelimesi gecen müşterileri bulmak için
"select * from müşteri where Adı like '%ve%'"
yeterli olacaktır. Peki bu en verimli yol mudur ?

Öncelikle su konu üzerinde biraz yoğunlaşalım. Bir Sql cümleciği veri tabanı yönetim sistemine gönderildiğinde önce bir parse işleminden geçmekte. Bir anlamda VTYS bizim ondan ne istediğimizi anlamaya çalışmakta. Bu 'anlama' işi bittikten sonra ise esas işleme başlamakta, yani kayıtları arayıp bulmakta.

Peki bu iki işlemden, yani, sql cümleciğini parse etme ve kayıtları aramak, parse etmek ile geçen zamanı azaltmamız mümkün mü ? Bu durumda gönderilen bir Sql'in sonucunu görme zamanı azalacaktır.

Bunun için TQuery'nin "Prepare" metodunu kullanabiliriz.
Örneğin ilk program yüklenirken:
Query1.Prepare;

Dememiz Query1'in sahip olduğu Sql cümlesinin parse edilmesini sağlayacak ve parse edilmiş halde hafızada kalacaktır. Burada dikkat edilmesi gereken bir konu, Query'in Sql'i değiştiği anda onu tekrar parse etmek ihtiyacının oluşacağıdır. Örneğin:
"select * from müşteri where adi like '%ve%'"
"selece * form müşteri where adı like '%al%'"
yukarıdaki iki sql'in amacı aynı olmasına rağmen veriler farklı olduğu için her seferinde tekrar parse işlemi yapılacaktır.
Bunun önüne parametre kullanarak geçebiliriz.
Örneğin Query1'in sahip olduğu sql her zaman
"select * from müşteri where adı like :adı" olsun.
Program ilk açıldığında Query1'i parepare edelim.
Kullanıcının aramak istediği metne göre yapmamız gereken tek şey:
Query1.Close;
Query1.ParamByName('adi').AsString := '%' + Edit1.Text + '%'
Query1.Open
olacaktır. Unutmayın bir query'nin kapatılması onun sql'inin prepare edilmiş halini yok etmez, tekrar prepare işlemi gerektiren tek durum Sql'in değişmesidir.
Yukarıdakileri özetleyecek olursak;

Esas amacımız bir Sql cümleciğinin sonuçlarını görme zamanını olabildiğince azaltmak. Bunun için bir anlamda TQuery'nin sahip olduğu Sql'i hiç değiştirmiyoruz ve 'compile' edilmiş olarak hafızada tutuyoruz. Bu bize o Query'i her çalıştırdığımızda parse işlemiyle gecen zamanın yol edilmesini sağlıyor.

Bu yolla programlarınıza ciddi anlamda hız artısı sağlayabilirsiniz. Programlar bölümüne bu işlemle ilgili bir örnek program koydum. İncelemenizi öneririm

2.3.2.TQuery veya TTable , Hangisini Kullanmalı

Programcılara Delphi veriyi acmak için TTable ve TQuery sınıfından iki adet bilesen sunmaktadır.Peki bu iki bileşenden hangisi seçilmeli, hangi durumlarda hangisi daha verimli sonuç verecektir ?

Öncelikle her iki bilesen de TBDEDataset sınıfından türemiştir. Bu da su anlama gelmektedir, her iki nesne de veriyi alırken BDE'nin avantajlarını ve daha çok dezavantajlarını kullanmaktadır.

TTable'i daha çok Paradox veya DBase gibi lokal veri tabanlarıyla kullanmayı tercih ediniz. Bunu önermemin sebebi Paradox ve Dbase'in aslında veri tabanı yönetim sistemi (VTYS) olmaması, bu isin BDE tarafından yapılmasıdır. Bu durumlarda TQuery kullanmanız size büyük dezavantajlar getirebilir.Paradox ile TQuery kullanmanın oldukça yavaş olmaktadır.

Eğer VTYS (Oracle, Interbase vs.) kullanıyorsanız veri için size e TQuery kullanmanızı öneririm. TTable bu gibi durumlarda basınıza büyük dertler açabilir. Bu dertlerden bir örnek, TTable'in VTYS'ne gönderdiği SQL'i değiştirememenizden kaynaklı olarak çok büyük performans düşüklükleri olabilir.

TTable daha çok basit, çok fazla karışık olmayan uygulamalar için geliştirilmiştir. Kullanıcının Tablo adı ve veri tabanını seçmesi TTable için veriye ulaşmakta yeterli bir bilgidir. Elbette bu kolaylik beraberinde dezavantajlar da getirmektedir. Buna bir ornek kayit aramalarini verebilirim. Bunun icin TTable'in filtreleme ozelligini kullandiginizda filtreleme islemi tamamen BDE tarafindan yapilacaktir. Bu ise su anlama gelmektedir, VTYS ne kadar hizli olursa olsun BDE'nin hizina mahkumsunuz.

Her zaman için uygulamanıza ve sisteminize göre TTable ve TQuery konusunda yapacağınız seçim, sisteminizin güvenliğini ve performansını büyük ölçüde etkileyecektir.

2.4. TStoredProc

Bu bileşen SQL tabanlı veritabanları üzerinde yazdığınız prosedürleri veya fonksiyonları kullanmanızı sağlayan bir bileşendir. Eğer Oracle, Sysbase, SQL Server gibi veritabanı kullanmıyorsanız bu bileşene ihtiyacınız yok demektir. Fakat adı anılanlardan herhangi biri veya bunlara benzer bir veritabanı sistemi kullanıyorsanız bu bileşen çok işinize yarayacaktır. Avantajları temel olarak şunlardır. Bu çalıştıracağınız prosedürler veritabanı üzerindedir. Normal kullandığınız query'lere göre çok daha hızlı çalışırlar. Network trafiğini asgari seviyeye indirirler. Yazacağınız prosedürle ilgili kullandığınız veritabanının dökümatasyonuna bakmalısınız.

Örn: Prosedür şöyle olsun.

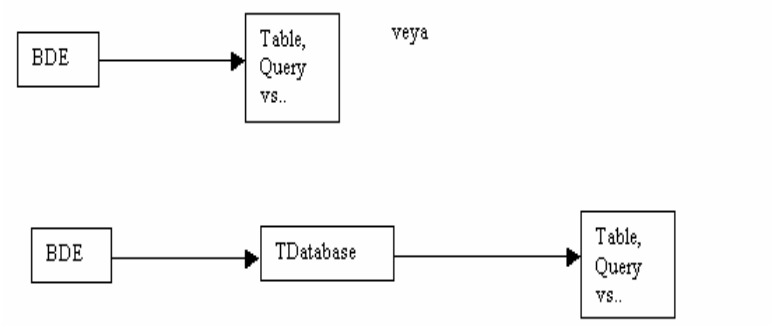
```
PROCEDURE  TEST (ISIM OUT VARCHAR2, NUMARA NUMBER)  IS
BEGIN
SELECT ADI INTO :ISIM FROM ABONE WHERE ABONE_NO=:NUMARA;
END;
```

Bunu Delphi'de aşağıdaki gibi kullanabilirsiniz;

```
StoredProc1.ParamByName('Numara').Value:='1000';
StoredProc1.GetResults;
Abone_Ismi:= StoredProc1.ParamByName('Isim').AsString;
```

2.5. TDatabase

Bu bileşeni projeniz içinde yönetim kolaylığı sağlamak için kullanabilirsiniz. Mesela TEST diye BDE alias'ınız var. Projedeki bütün table'lar query'ler vs. hepsi buna bağlı. Bunun ismini değiştirdiğinizde bütün projede gidip aliasları değiştirmeniz gerekir. Bunun yerine TDatabase bileşenini kullanırsanız, sadece bu bileşenin alias'ını değiştirdiğinizde projede buna bağlı ne kadar bileşen varsa hepsini etkileyecektir.



Şekil 2.3 Tdatabase

şeklinde olabilir.

2.6. TDataSource

Verilerinizin data controls sayfasındaki bileşenler yardımı ile görüntülenmesi için table,query vb. gibi datasetleri mutlaka bir **DataSource**'a bağlamanız gerekir.

2.7. Data Controls

TDBGrid: Verilerin gösterilmesi için kullanılır. Gösterilecek alanlar ayarlanabilir, verilerin fontu ve grid'in başlık fontu ayrı ayrı ayarlanabilir.

TDBNavigator: Veriler üzerinde güncelleme, silme, yeni kayıt, ileri-geri gitme vs. gibi işlemlerin yapıldığı araç çubuğudur. Bunu bir datasource'a bağladığınız zaman bu datasource'un bağlı olduğu dataset navigator de yaptığınız tüm işlemlerden etkilenir.

TDBText: Label ile aynıdır. Fakat bağlı olduğu tablodan belirtilen alan bilgisini görüntüler. Genelde üzerinde değişiklik yapılmayacak alanların gösterilmesi için uygundur.

TDBEdit: DBText ile benzerdir, ek olarak eğer imkan dahilinde ise veriler üzerinde değişiklik yapılabilir.

TDBMemo: Birden fazla satırın yada 255 karakterden daha uzun verilerin saklanması ve gösterilmesi için kullanılır.

TDBImage: Veritabanlarında resim içeren alanların işlenmesi için kullanılır.

TDBListBox: Bunun özelliği şudur. Verdiğiniz alan değeri eğer listeni içinde bulunuyorsa otomatik olarak seçilir. Bileşenin listesini siz kendiniz doldurmak zorundasınız. Verdiğiniz alan ile ilgili değerler otomatik olarak gelmez. Mesela listesi Table1'in İsim alanına bağladınız. Etkin kayıttaki isim 'Ahmet' ve liste elemanları içinde de 'Ahmet' varsa bu eleman otomatik olarak işaretlenir. Aksi halde yani liste elemanları içinde 'Ahmet' yoksa listede hiçbir eleman işaretlenmez. Büyük-küçük harf fark etmez.

TDBComboBox: TListBox ile aynıdır. Eğer elemanları arasında etkin kaydın ilgili değeri varsa görüntülenir, diğer halde boş olarak gözükür.

TDBCheckBox: Genelde iki durumlu veriler için kullanılır. Mesela, personel tablosu için düşünecek olursanız kişi yönetici veya değil. Bunun için kullanılabilir. Yönetici olması 1, olmaması 0 ile gösterilecek olursa **ValueChecked**=1, **ValueUnChecked**=0 olmalıdır. Eğer veriyi sadece göstermek amacı ile bu bileşeni göstermek kullanmak isterseniz o zaman **ValueChecked ve ValueUnChecked** özelliklerine birden fazla değer girebilirsiniz. Mesela 2 üst düzey yöneticiyi göstersin. Eğer detay gerekmiyorsa yönetici olduğunu göstermek için ValueChecked=1;2 şeklinde değer girebilirsiniz.

TDBRadioGroup: Bu bileşen tablolarda kod olarak tutulan fakat programda açık olarak yazılması gereken alanlar için kullanılır. Mesala, 0-İşçi, 1-Yönetici, 2-Üst düzey yönetici ise bu bileşenin Items özelliğine alt alta İşçi, Yönetici ve Üst Düzey Yönetici, Values özelliğine de yine alt alta 0,1 ve 2 yazarsanız veritabanında 0 olduğu zaman İşçi, 1 olduğu zaman Yönetici ve 2 olduğu zaman Üst Düzey Yönetici otomatik olarak RadioGroup içinde seçilecektir. Aynı bunları yeni veri girişi yaparken de kullanabilirsiniz.

TDBLookupListBox: Bu bileşen başka bir tablonun belirtilen alanında bulunan tüm verileri görmek ve aynı zamanda istenirse bu verilerle ilgili bilgileri başka bir tabloya kaydetmek için kullanılır. Örn: Personel tablonuzda personelin oturduğu il kodunu tutuyorsunuz. Bunun için birde IlKodu,IlAdı'nı içeren Iller tablonuz var. O zaman bu bileşenin Datasource'u olarak personel, datafield'i ilkodu, ListSource'u ıller, listFields'ı il_adi ve keyField'i da ıl_kodu verdiğiniz zaman illerden Ankara'yı seçtiğiniz zaman personelin ilkodu alanına '06' otomatik olarak yazılacaktır. Aynı şekilde ilkodu '34' olan bir personelde Illerden İstanbul otomatik olarak seçilektir.

TDBLookupComboBox: TDBLookupListBox ile aynıdır.

TDBRichEdit: Eğer veritablarında 64Kb'den daha büyük bilgini işlenmesi gerekiyor bu bileşen kullanılır.

TDBCtrlGrid: DBGrid gibidir. Fakat bunda her bir kayıtın için gösterilicek alanları, bu olanların nasıl gösterileceğini (herbiri ayrı bir renkte, farklı büyüklükte gibi) belirleyebilirsiniz.

TDBChart: Veritabanındaki verilerden otomatik grafik oluşturmak için kullanılır.

3. ISAPI/CGI TABANLI WEB UYGULAMALARI

3.1. Isapi/Cgi'ye Giriş

Web (HTML) sayfaları aslında durağan yani statik sayfalardır. Bu sayfalara dinamizm katmanın pek çok yolu bulunmaktadır.Mesela bunlar ASP (Active Server Page), ISAPI (INTERNET SERVER APPLICATION PROGRAMING INTERFACE), CGI (COMMON GATEWAY INTERFACE) uygulamalarıdır.Biz burada daha çok İsapi arabirimi üzerinde yoğunlaşacağız.

3.2. Isapi/Cgi Ve Diğer Delphi Uygulamaları Arasındaki Fark Nedir

ISAPI bir DLL uygulamasıdır. CGI ise bir konsol (Windows için konuşursak penceresi olmayan çıktısını bir tarayıcıya örneğin IE. yönlendiren) bir EXE uygulamasıdır. Delphinin diğer uygulama standartları ile bu iki tur uygulamayı birbirinden ayıran yegane fark, tüm Delphi uygulamalarında kullanıcı bir Windows OS penceresi üzerindeki denetimler ile çalışırken, ISAPI ve CGI uygulamalarında HTML üzerindeki denetimlerle çalışılmasıdır. ISAPI DLL olmasından ötürü çalıştırıldığında özel olarak serbest bırakılmadığı surece belleğe bir kez yüklenir ve her sefer çağrılısında ayni bellek adresinden koşar. CGI ise bir exe olmasından dolayı her çağrılısında belleğe yeniden yüklenir ve isi bitince ISAPI nin aksine sonlanır.
Tahmin edebileceğiniz gibi ISAPI CGI uygulamalarına nazaran çok daha hızlı çalışır, ancak herhangi bir istisna çıktığında çalıştığı ana bilgisayarı dolayısıyla tüm sistemi çökertebilir.

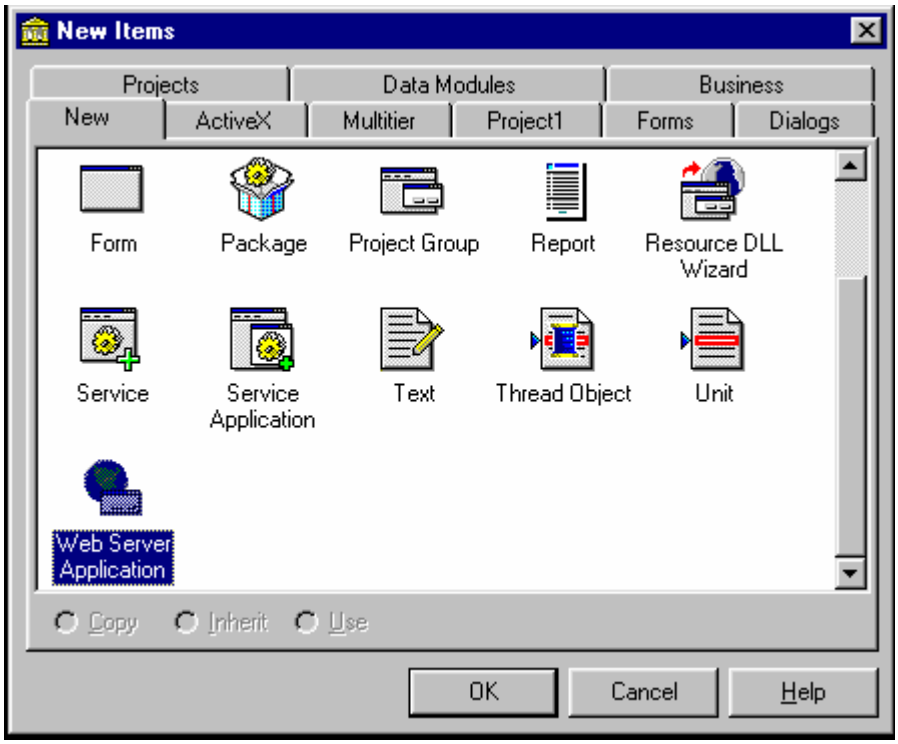
3.3. Isapi/Cgi Uygulamaları Nasıl Yazılır

ISAPI VE CGI arasındaki farkı öğrendiğimize göre bu tur uygulamaların nasıl yazılacağı konusuna bir giriş yapabiliriz. Aşağıdaki işlem basamaklarını sırasıyla izleyelim.

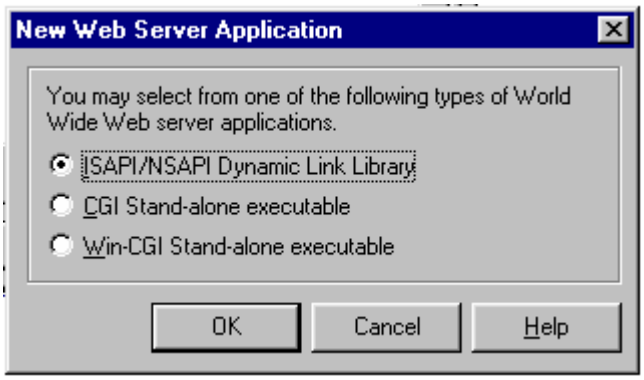
1- Delphi ana menüden New secin.

2- New Items penceresinin en altında bulunan Web server application item'i secin(Şekil 3.1)

3- Karsınıza gelecek pencerede 3 adet seçenek var. 1.si İSAPİ/NSASPİ 2si CGİ ve 3 ncu seçenek CGI uygulamalarının veri iletimi için Windows Ini sistemine benzer bir sistemi kullanıyor. (Şekil3.2)



Şekil 3.1 Web Server Application

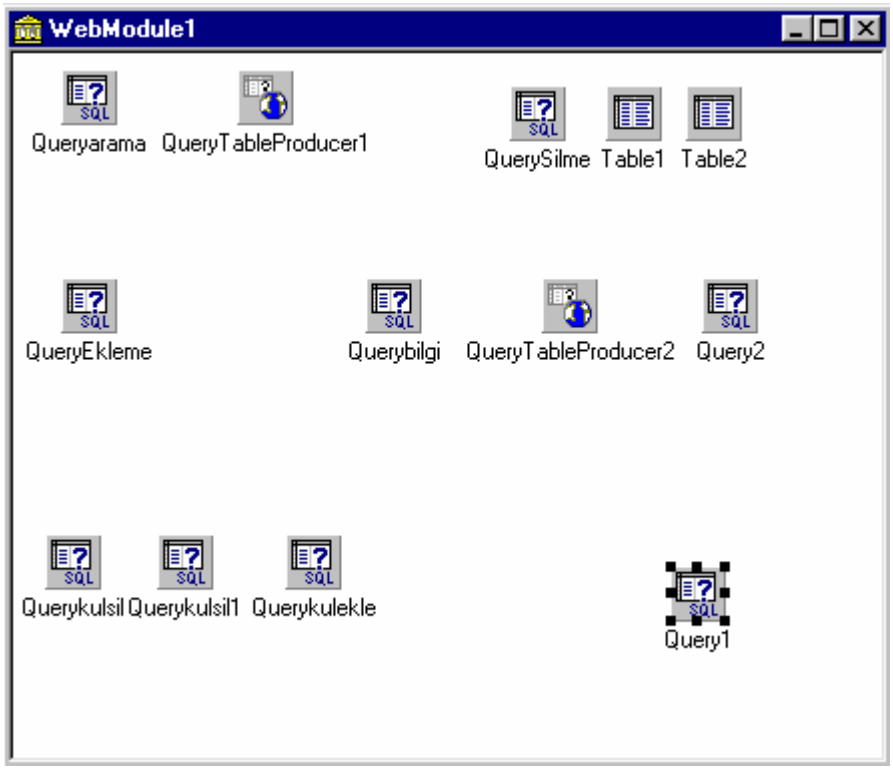


Şekil 3.2 New Web Server Application

NOT : ISAPI ve CGI uygulamalarının genel yapısı ve kullandıkları bileşenler aynidir. Daha sonrada göreceğimiz gibi bir ISAPI uygulamasını proje kaynak dosyasındaki Bir kaç satiri silip library ifadesi yerine program yazarak veya tersini yaparak bir türden diğerine geçebiliriz.

4- Karsınıza şekil 3.3 teki gibi bir WEBMODULE çıkar.

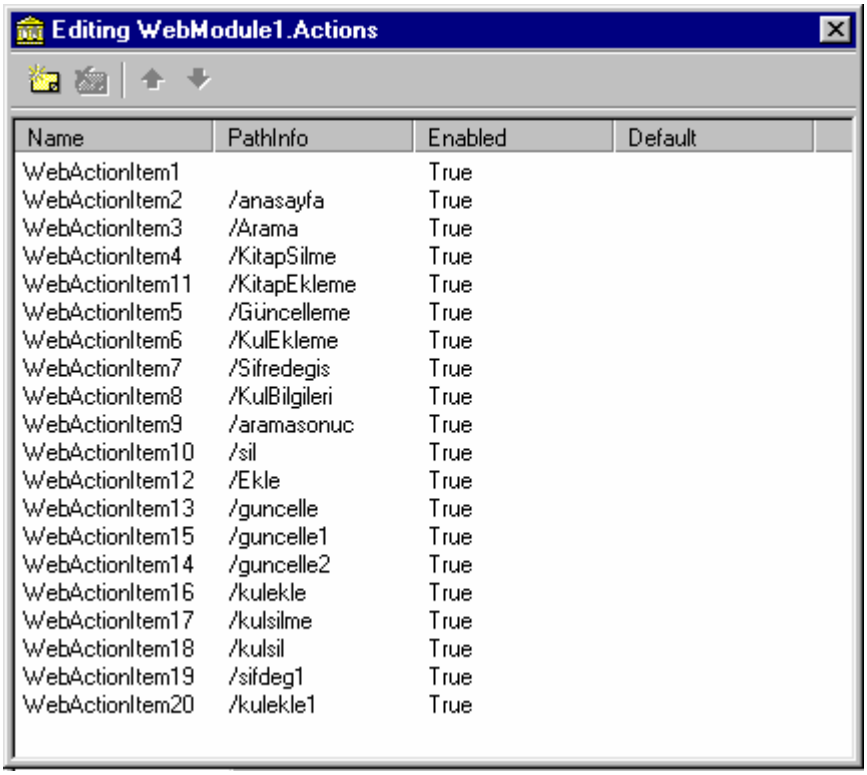
3.4. Webmodule



Şekil 3.3 TwebModule

WebModule için help'e giderseniz TDataModule sınıfından türeyen bir nesne olduğunu görürsünüz. Sanırım en azından bir programınızda DataModule kullanmışsınızdır. İste tıpkı datamodule nesnesinde olduğu gibi WEBMODULE nesneside TDataset (TQuery, TDataBase,TTable vs.) bileşenlerine konteynırlık görevini üstlenmektedir. Ama DataModuleden farklı olarak HTML sayfaları ile iletişim kurabilen çok önemli bir özelliği bulunmaktadır. (Aslında bir DataModule üzerine bir WebDispatcher:TWebDispatcher bileseni koyarsaniz Datamodule nesnenizi Webmodule olarak kullanabilirsiniz).

Bir Webmodule'un en önemli özelliği üzerine çift tıkladığınızda veya sağ tıklayarak açtığınızda acılan WebActionItem.Editor ile düzenleyebildiğiniz Action lardir.. Bir webmodule HTML ile ilgili iki işlem yapar.



Şekil 3.4 Webmodule Actions

1- Talepleri almak (Request:TWebRequest)
2- Talepleri HTML'ye göndermek (Response : TWebResponse)
Twebactionitem1 için bir olay yöneticisi yazmak için action editörde webactionitem1 ‘i işaretledikten sonra events sekmesine geçip ve onaction olayına çift tıkladıktan sonra Aşağıdaki gibi bir kodla karşılaşırız.

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
end;
```

Burada öğrenmemiz gereken respose ve request parametreleridir.

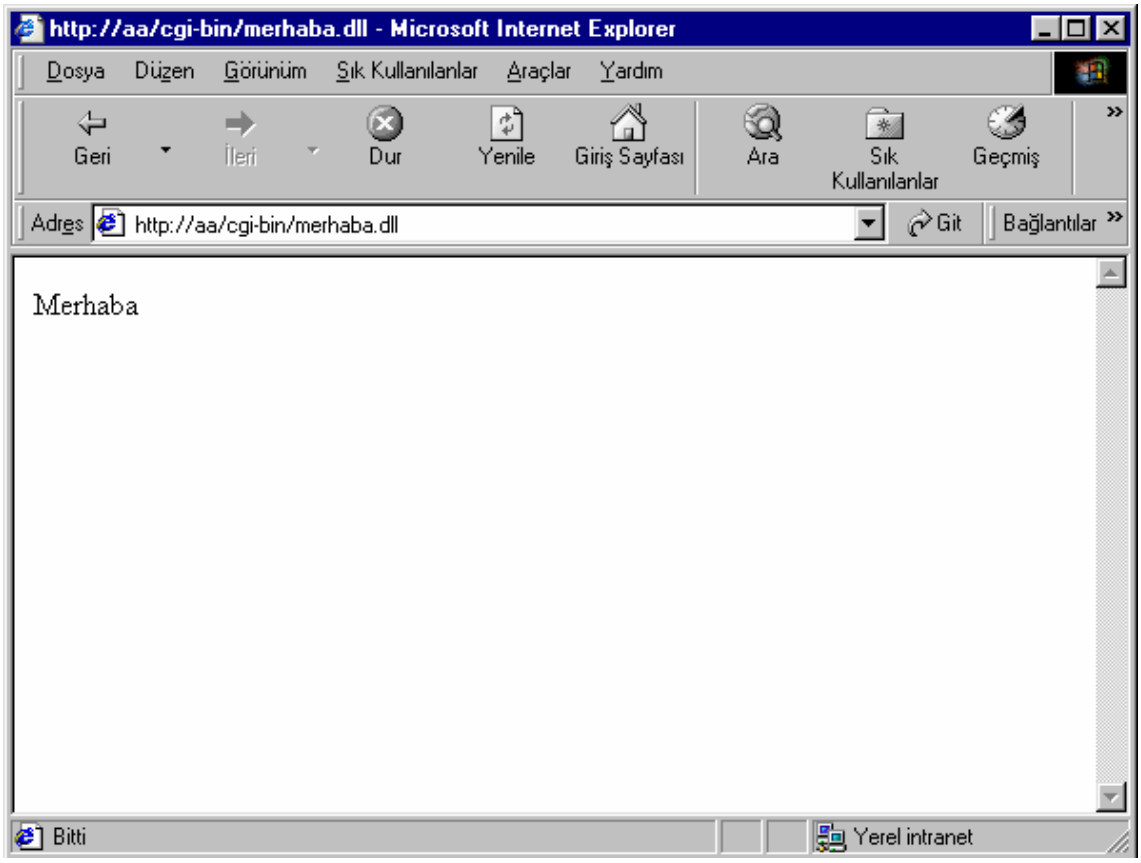
3.4.1. Twebresponse

Response çıktının ne olacağını belirlemek için kullanılır.response parametresine html yerleştirebilmemiz için content özeliliğini kullanmamız gerekir. Örneğin Aşağıdaki kod “Merhaba” çıktısını verir.

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;

Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
response.content:='<H1>Merhaba</H1>'
end;
```

Bu kodu yazdıktan sonra programı merhaba.DPT şeklinde kaydedelim.Bundan sonra programı derlemeliyiz.Derleme sonucunda programımızın otomatik olarak dll’i oluşturulacaktır(merhaba.dll).Bizim görevimiz sadece bu dll dosyasını alıp ; personal web server kullanıyorsak cgi-bin dizinine , ISS(internet information server) ‘in scripts dizinine tasımalıyız.Bundan sonra sadece internet explorer’ın adres satırına dosyanın yollunu yazmamız yeterlidir.



Şekil 3.5 Twebresponse

Burada yazacağımız adres satırı PWS kullanıyorsak şu formatta olmalıdır:

http://bilgisayar_adı/cgi-bin/merhaba.dll

ISS kullanılıyorsa

http://bilgisayar_adı/scripts/merhaba.dll

3.4.2. Twebrequest

Request sorgusunun birkaç önemli özelliği vardır.Sorguyu göndermekte kullanılan yönteme bağlı olarak queryfields veya contentfields kullanılabilir.

HTML konusunda bilmeniz gereken bir kaç noktaya dikkat etmek lazımdır. Bunlar ;

1- HTML dosyaları statiktir

2- HTML dosyaları tarayıcıya yüklenirken ilk satırdaki taglardan başlayıp, satır satır tarama yapılarak sayfa yüklenir

3- Bir HTML'nin bir ISAPI/CGI uygulamasına talep/data göndermesi için FORM tabi kullanılır.

```
<Form Action="http://aa/cgi-bin/.kutuphane.dll"
method="post" encoding="x-www-formencoded">

<input type="text" name="adı" value="">
<input type="text" name="soyadı" value="">
<INPUT type="submit">
</form>
```

Yukarıdaki örnekte bir web sayfası içine ismi bilgial.DLL olan ve tam yol ismi (Adresi) verilen ISAPI uygulamasına adı ve soyadı alanlarının değerleri gönderilmektedir.

NOT : Bilmeyenler için; HTML'de veri gönderirken en çok kullanılan iki method vardır. POST ve GET yöntemleri. POST metodunu kullandığınızda veri boyu bilgisayarınızın kapasitesi ile doğru orantılıdır, gönderdiğiniz bilgide kullanıcı tarafından görülmez. GET yöntemi ile veri göndermek post metoduna göre oldukça hızlıdır, ancak gönderebileceğiniz mam. veri 250 karakterdir ve gönderilen veri tarayıcınızın adres alanında kullanıcı tarafından görülebilir.Tabii ki Delphiden gönderilen bilgiye erismek için POST ve GET yöntemlerinden kullanılan methoda uygun Delphi yöntemini kullanmanız gerekmektedir.

Bir html sayfası ile bir isapi/cgi uygulamasının etkileşimde bulunabilmesi için <Form> html tagının kullanılır ;

```
<HTML>
<HEAD><TITLE>DELPHI ISAPI-CGI UYGULAMALARI </TITLE><HEAD>
<BODY>
<!—kitap adı.dll'e post metodu ile veri gonderimi -->
<Form Action="http://aa/cgi-bin//kutuphane.dll"
method="post"
KitapAd:<input type="text" name="kadı" value="">
Yazarı:<input type="text" name="yazarı" value="">
<INPUT type="submit">
</form>
</BODY>
```

</HTML>

Bir Webmodule:TWebModule nesnesinin en önemli özelliği WebActionItem : TWebActionItem özelliklerdir.

3.4.3 TWebActionItem

Bir WebActionItem oluşturmak için ya webmodule üzerinde çift tıklayarak yada kıs ayol menusu ile (mouse sağ tuşa tıklayıp)

Action Editor penceresini acın ve Add seçeneğini kullanın. Bir webmodule bir yada daha fazla sayıda WebActionItem ekleyebilir ve bunların özellik ve OnAction olayını object inspector ile ayarlayabilirsiniz. İste bu özelliklerden önemli olanların ve tek olayın (ONACTION) detayları ;

3.4.4.Özellik/Olay (Onaction)Açıklama

Default ilgili WebActionItem'in bir pathinfo kullanılmadan HTML sayfalarından gelen / gonderılen talep veya değerlerin varsayılan yanıtlama olayı olarak ayarlanması. Yukarıdaki örnekte bir pathinfo belirtilmemiştir.Dolayısıyla Webmodule default özelliği true olan WebActionItem'in onAction olayı ile talebe tepki verecektir.
Methodtype HTML üzerinden gönderilen veri/talebin gönderilme biçimi. Varsayılan değeri mtAny. Bir html sayfası form tagı ile isapi/cgi uygulamaya değer gönderirken 4 yöntemden birini kullanır.

Pathinfo yukarıda belirttiğim gibi bir Webmodule ait birden fazla WebActionItem olabilir. Eğer birden fazla WebActionItem kullanırsanız, html sayfaların-dan bu WebActionItem'lara ulaşmak için spesifik birer isim vermeniz ve Form tagına bunu eklemeniz gerekir.

```
<!-- kütüphane.dll'e post metodu ile kullanıcı      WebActionItem na veri gonderimi -->
<Form " http://aa/cgi-bin/kütüphane.DLL/kullanıcı "
method="post" >
Adınız : <input type="text" name="adı" value="">
Soyadınız : <input type="text" name="soyadı" value="">
<INPUT type="submit">
```

</form>

Producer herhangi WebActionItem'a bir Pageproducer veya bunun türevlerini atamak için kullanılır.

Bir WebActionItem bir producer atarsanız talebin yanıtlanması ile ilgili sonuç çıktısı bu producer ile gosterdiginiz kaynak tarafından yapılır.

OnAction(Olay): Bir html sayfası ile etkileşim kuran webmodule bunu WebActionItem OnAction olayı ile değerlendirir.Değerlendirmeden kastım, gelen talebi alır, yazdığınız kod doğrultusunda isler (Bir veri tabanı kayıt işlemi yapmak gibi), ve yine bu olaydaki kodlar doğrultusunda üretilen html çıktısını talebi gönderen bilgisayarın Webbrowserina geri gönderir.Tüm bu bilgiler ışığında bir html sayfası ile bir isapi/cgi uygulaması arasındaki trafiği aşağıdaki biçimde özetleyebiliriz.HTML SAYFASI : Form tagını kullanarak uygulamadan (isapi/cgi) talepte bulunur.

<FormAction="http://aa/cgi-bin/kütüphane.dll/yazarı "
method="post>
Component Turu : <input type="text" name="tur" value="">
<INPUT type="submit">
</form>

3.5. ISAPI/CGI uygulaması

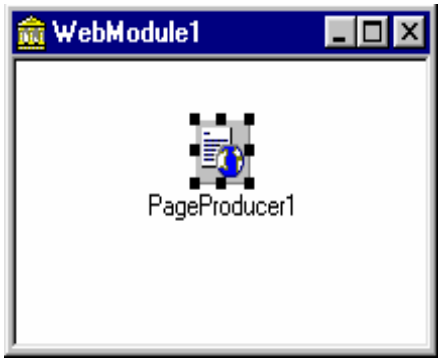
Html sayfasından gelen/ gönderilen talep/veriyi kabul eder. OnAction olayı ile isler ve istenirse geriye talep veya işlemin sonuçlarını dondurur. Talebi kabul ederken Form tagında gösterilen Form Action="http:// http://aa/cgi-bin/kütüphane.DLL/giris" satırında gösterilen pathinfo yu dikkate alır. (Örnekte pathinfo=componentsuz) Bu Pathinfoya ait WebActionItemin ONAction olayında yazılan kodlar isletilir.

3.6. TPageproducer

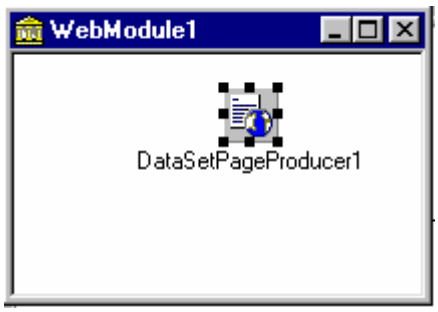
Response.content özelliğinden dolayı istersek sayfamızın tüm html kodlarını bu değişkene yükleyebiliriz fakat bazen belirli alanların(bir isim veya tarih bilgisinin) doldurulması gerektiğinde bir şablona dayalı html katarı döndürmek isteyebiliriz.Bu durumda Tpage producer bileşeni kullanmamız gerekir. (Şekil 3.6)

3.7. Tdataset pageproducer

Tdataset pageproducer bileşeni Tpageproducer bileşeninden üretilmiştir.Tdataset pageprodu cer bileşeni sadece # imlerini düzenli bir değerle değiştirmek yerine yeni bir dataset özelliğine sahiptir ve # iminin adını ,dataset özelliğinin bir fieldname 'i ile eşlemeye hazırdır. Tdataset pageproducer böyle bir tane bulursa # imini bu alanın asıl değeriyle değişir. (Şekil 3.7)



Şekil 3.6 Tpageproducer

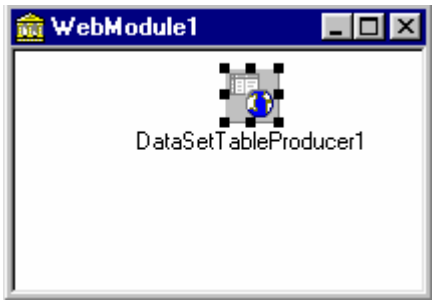


Şekil 3.7 Tdatasetpageproducer

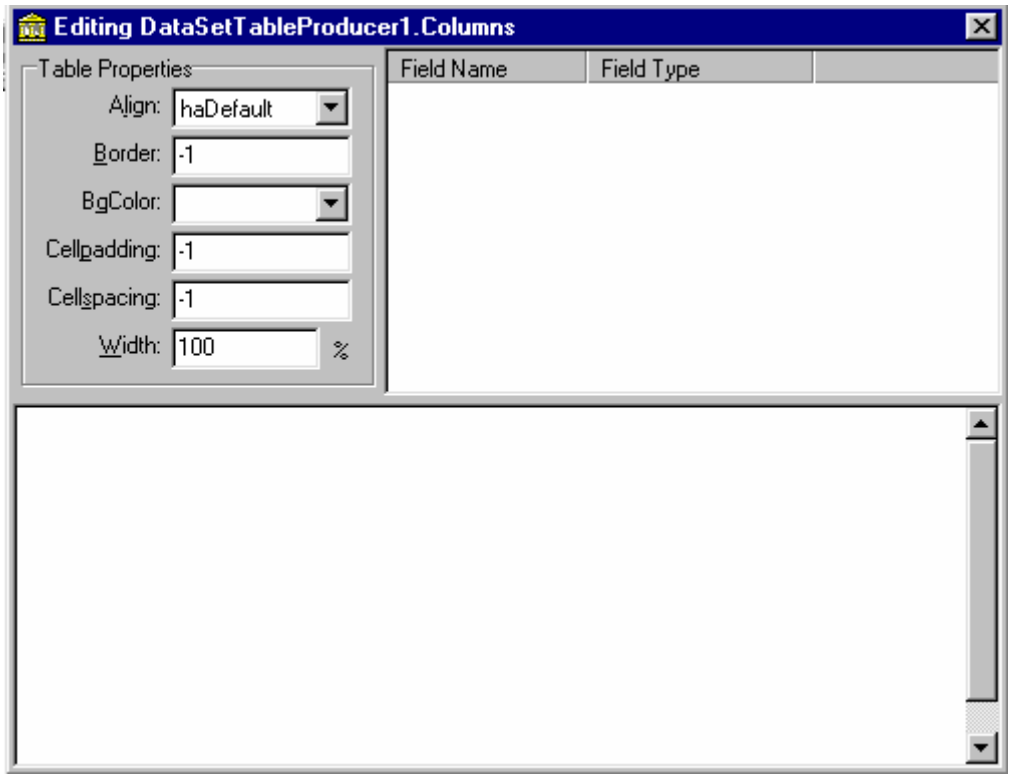
3.8. Tdatasettableproducer

Tdatasettableproducer bileşenide Tdatasetpageproducer bileşeni gibi dataset özelliğini kullanır.Ancak bu sefer birden fazla kayıt elde edersiniz , üstelik çıktı ağa benzer bir tablo biçimindedir. (Şekil 3.8)

Datasettebleproducer'a çift tıkladığımızda karşımıza aşağıdaki gibi bir şekil çıkar.Burada veritabanı tablolarımızın web sayfamızda nasıl görüneceği ile ilgili ayarlamaları yapabilmekteyiz. (Şekil 3.9)



Şekil 3.8 TdatasetPageproducer



Şekil 3.9 Editing datasettableproducer colums

3.9. Tquerytableproducer

Tquerytableproducer tdatasettableproducer bileşenine benzer bir çıktı üretir.Aralarındaki fark Tquerytableproducer bileşenine sadece query bağlayabilmemiz değil,Tdatasettableproducer bileşeninin parametrikleştirilmiş bir tquery parametrelerini doldurmak için özel bir desteği olabilmesidir.

Şimdi yukarıda anlatılanların proje içerisinde nasıl uygulandığını göstermek için projeye ait bir alt modülün nasıl inşa edildiğini anlatmaya çalışacağım.Bunun için Kitap Arama altpogramının nasıl tasarlandığını anlatmaya çalışacağım.İlk önce sayfanın tasarımını yaptım. Sayfanın görünümü aşağıda gösterilmektedir.

Şekil 3.10 Kitap Arama

Burada kullanıcı sayfada bulunan kriterlerden istediği birine göre arama yaptırabilmektedir.Bu kriterler;Kullanıcı_Adı ,Kitap_ismi,Yazarı,Basım_yılı,Yayın_evi konusu şeklindedir.Kullanıcı bu alanlardan istediği birini doldurarak arama yapabilmektedir.

Sayfann tasarımından sonra sql veritabanından arama yapacağım için arama yapmamı sağlayacak sorguyu webmodule yerleştirdim. Şekil3.3 teki queryarama sorgusunu kullanmaktayım.Bu query nin içeriği şu şekildedir.

```
select * from  kitap
where  kullanıcı_adı=:kullanıcı_adı
or  kitap_ismi=:kitap_ismi or
yazarı=:yazarı or basım_tarihi=:basım_tarihi
or Yayın_evi=:yayın_evi
or konusu=:konusu
```

Burada kitap arama yapacağım tablonun adı, kullanıcı_adı,kitap_ismi basım_tarihi,yazarı,yayın_evi,konusu arama yapmamda yardımcı olacak parametrelerdir bu parametreler aynı zamanda kitap veri tabanındaki entitiylerdir.Parametrelerin türünü params seçenegi ile ayarladım.
Bundan sonraki adımda Şekil3.4 de bir webactionitem oluşturdum ve objectinspectorda pathinfosuna /arama adını verdim.Yani bunun anlamı kullanıcı Web sayfasındaki kitap arama linkine tıkladığında /arama sayfasına gider.Bu ayarlamaları yaptıktan sonra /arama webactionun onactions olayına (object inspectorda events bölümünde) aşağıdaki kodu yazdım.

```
procedure TWebModule1.WebModule1WebActionItem3Action(Sender: TObject;
Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
response.content:=response.content=:+'<html>'+''+'<head>'+'<style><!-+
'a:link{ color:yellow; font-size:10pt; font-family:tahoma; text-decoration:underline; }'+
```

```
'a:hover{ color:yellow; background:red; font-size:10pt; font-family:verdana; text-decoration:none; }'+
'a:visited{ color:yellow; font-size:10pt; font-family:verdana; text-decoration:underline; }'+
'-->'+'</style>'+'<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">'+
'<title>AnaSayfa</title>'+'<metaname="GENERATOR"3.0">'+
'</head>'+"+'<bodybgcolor="#000080">'+
'<div align="center"><center>'+'
'+
'<table border="0" width="870" height="91" bgcolor="#800000">'+'
 <tr>'+
'<td width="155" valign="top" rowspan="2" bgcolor="#008080" height="91"><br>'+
'<p> </p>'+'<p><ahref="http://aa/cgibin/kitos.dll/anasayfa"><strong>AnaSayfa</strong></a></p>'+'<p><ahref="http://aa/cgibin/kitos.dll/Arama"><strong>Kitap Arama</strong></a></p>'+
'<p><fontcolor="#FFFFFF"><a href="http://aa/cgi-bin/kitos.dll/KitapSilme"><strong>Kitap Silme</strong></a></font></p>'+
'  <p><font color="#FFFFFF"><a href="http://aa/cgi-bin/kitos.dll/KitapEkleme"><strong>Kitap Ekleme</strong></a></font></p>'+
'  <p><font color="#FFFFFF"><a href="http://aa/cgi-bin/kitos.dll/Güncelleme"><strong>Kitap Güncelleme</strong></a></font></p>'+
'  <p><font color="#FFFFFF"><a href="http://aa/cgi-bin/kitos.dll/KulEkleme"><strong>Kullanıcı Ekleme</strong></a></font></p>'+
'  <p><font color="#FFFFFF"><a href="http://aa/cgi-bin/kitos.dll/Kulsilme"><strong>Kullanıcı Silme</strong></a></font></p>'+
'  <p><font color="#FFFFFF"><a href="http://aa/cgi-bin/kitos.dll/Sifredegis"><strong>Sifre Degiştirme </strong></a></font></p>'+'  <p> </p>'+
'  <p> </td>'+'  <td width="703" height="511" valign="top" align="center"><p align="left"><font'+
'  color="#00FFFF" size="5"><strong>KİTAP ARAMA</strong> </font></p>'+'  <p align="left"> </p>'+
```

```
'            <p align="left"><font color="#00FFFF" face="times New Roman" size="3"><strong>Aşağıdaki'+
'    kutucuklardan herhangi birini doldurarak istediğiniz kriterde arama yaptırabilirsiniz </strong></font></p>'+   <p align="left"> </p>'+
'    <form action="http://aa/cgi-bin/kitos.dll/aramasonuc" method="post">'+'      <table border="0" width="59%" height="150">'+
'      <tr>'+
'              <td width="25%" height="19" align="right"><div align="left"><p><font color="#00FFFF"> '+
'        Kullanıcı Adı  </font></td>'+
'               <td width="75%" height="19"><input type="text" name=Kullanıcı_adı size="23"></td>'+
'      </tr>'+'
       <tr>"                    <td width="25%" height="19" align="right"><div align="left"><p><font color="#00FFFF"> '+'          Kitap İsmi   </font></td>'+
'                 <td width="75%" height="19"><input type="text" name=Kitap_ismi size="23"></td>'+
'      </tr>'+'      <tr>'+
'        <td width="25%" height="19" align="right"><div align="left"><p> <font color="#00FFFF">'+
'                  Yazarı   </font></td>'+'                <td width="75%" height="19"><input type="text" name=Yazarı size="23"></td>'+
'     </tr>'+
'      <tr>'+
'             <td width="25%" height="19" align="right"><div align="left"><p><font color="#00FFFF"> '+'        Basım Yılı   </font></td>'+
'               <td width="75%" height="19"><input type="text" name=Basım_tarihi size="23"></td>'+
'      </tr>'+
'      <tr>'+
```

```
'                <td width="25%" height="8" align="right"><div align="left"><p><font color="#00FFFF"> '+
'                  Yayınevi    </font></td>'+'                <td width="75%" height="8"><input type="text" name=Yayın_evi size="23"></td>'+
'       </tr>'+
'       <tr>'+
'                <td width="25%" height="5" align="right"><div align="left"><p><font color="#00FFFF"> '+
'                  Konusu    </font></td>'+'                <td width="75%" height="5"><input type="text" name=Konusu size="23"></td>'+
'       </tr>'+
'</table>'+
'<divalign="left"><p>             '+'                '+'                '+
'                '+
'       <input type="submit" value="gönder" >  <input type="reset" value="Temizle"'+
' ></p>'+'   </div>'+'   </form>'+
'   </td>'+'  </tr>'+'  <tr>'+
'   <td width="703" valign="top" align="center" height="1"></td>'+
' </tr>'+'</table>'+'</center></div>'+'</body>'+'</html>'
end;
```

Burada görüldüğü gibi kod html kodlarından oluşmaktadır.Kodlar response.content ifadesinden sonra ‘ ‘+ işaretleri arasına alınarak derleyiciye bunların birer html kodu olduğu belirtilmektedir.Burada sayfada bulunan linklerin gideceği adreslerin atamaları yapılmaktadır.Burada önemli olan kodlardanlardan biri
<form action="http://aa/cgi-bin/kitap.dll/aramasonuc" method="post">'+kodudur.

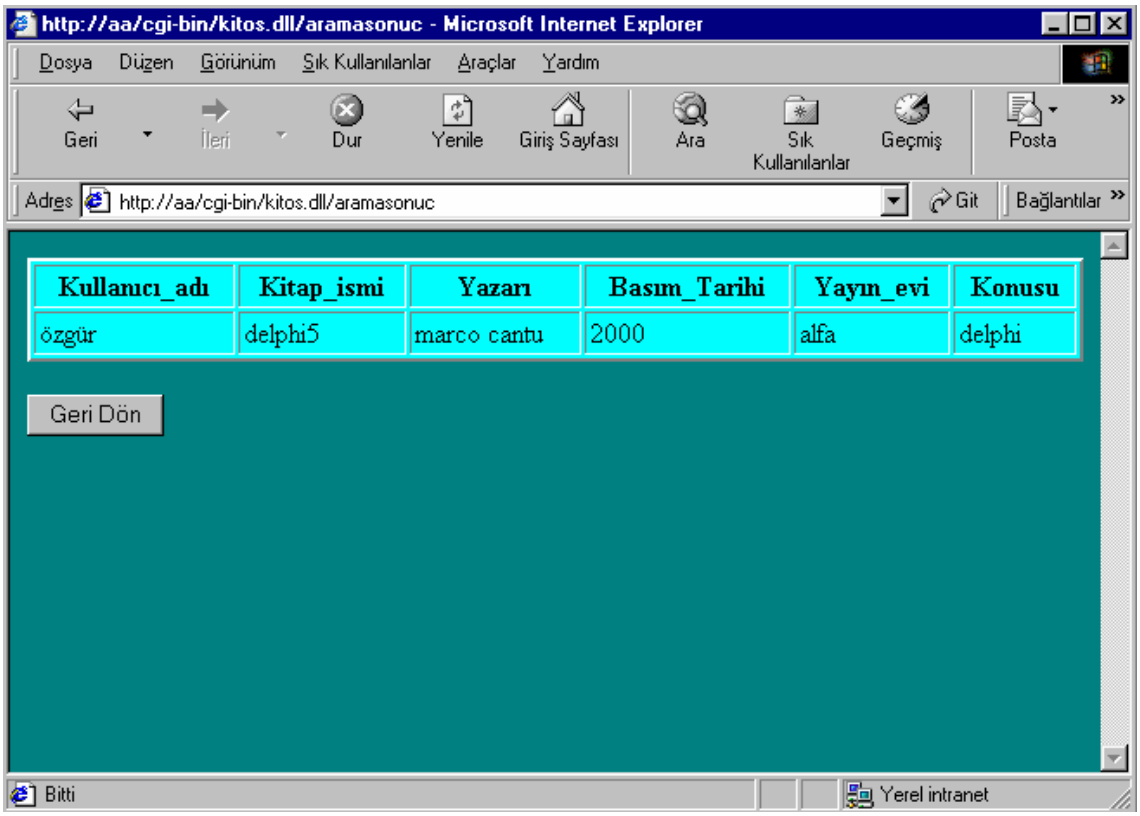
Bu kodla gönder butonuna tıklandığında alınan bilgilerle hangi sayfaya gidileceği daha doğrusu hangi webaction olayına gidileceği belirtilmektedir.Burdanda aramsonuc webactionunayani aramasonuc sayfasına gidileceği görülmektedir.Bu sayfanın kodunda dikkat edimesi gereken ikinci bir kod kısmı ise bilgilerin alınmasına dair olan kod kısmıdır.Buna örnek olarak su kodu gösterebiliriz ; <input type="text" name=Kitap_ismi > .Bu kod kısmi ile veritabanındaki Kitap tablosunda bulunan Kitap_ismi entitiy'si ile ilişki kurulmaktadır..Burada dikkat edilmesi gereken önemli bir nokta vardır.Edit 'in ismi ile tablonun entity 'si aynı olmalıdır.Çünkü bu edit içerisine yazacağımız değerle ilgili arama yapmaktayız.

Bundan sonraki kısımda ise arama sonuc sayfasınınınşası yer almaktadır.Asıl iş burada yapılmaktadır.web modulde buwebactionun action olayına aşağıdaki kod kısmını yazarız.

```
procedure TWebModule1.WebModule1WebActionItem9Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
Begin
queryarama.Close;
queryarama.params[0].asstring:=request.contentfields.values['Kullanıcı_adı'];
queryarama.params[1].asstring:=request.contentfields.values['Kitap_ismi']queryarama.p
arams[2].asstring:=request.contentfields.values['Yazarı'];
queryarama.params[3].asstring:=request.contentfields.values['Basım_tarihi'];
queryarama.params[4].asstring:=request.contentfields.values['Yayın_evi'];query arama.
params[5].asstring:=request.contentfields.values['Konusu'];
queryarama.open;response.content:=querytableproducer1.content
;response.content:=response.content +
'<html>'+
'<body bgcolor="#008080">'+
'<form method="POST" action="http://aa/cgi-bin/kitos.dll/Arama">'+' <p><input
type="submit" value="Geri Dön"></p>'+
'</form>'+
'</body> '+
'</html> ' end;
```

Şimdi bu kod kısmını biraz açıklamaya çalışacam.Burada ilk olarak webmodulünün üzerine yerleştirdiğim queryarama sorgusu kapatılmaktadır.Bunun nedeni query'mizin eğer daha önceden open metodu çağrılarak aktif hale getirilmişse kapanmasını sağlamaktır.Daha sonra query içinde kullanmış olduğumuz parametrelere arama sayfasından aldığımız değerleri yerleştirmekteyiz.Böylece veritabanında istediğimiz kriterlere göre arama yapabilmekteyiz.Burada istersek birden fazla kritere görede arama yaptırabiliriz.Arama sonuçlarını gösterilmesi için querytableproducer kullanılmıştır.

Aşağıda kitap_ismi kriterine göre yaptrdığımız arama sonucları gösterilmişitir.



Şekil 3.11 Arama Sonuç

Şekil 3.11'deki geridön butonu form action olayı ile bir önceki sayfaya gitmesi sağlanmıştır.

Sanal Kütüphanenin diğer işlemleri kitap ekleme,kitap silme ,güncelleme ,kullanıcı ekleme,kullanıcı silme işlemleri de biraz farklı olmakla beraber aynı mantıkla yapılmaktadır.Olayın anlaşılması için en kolay işlem olan arama işlemi anlatmaya çalıştım.

Bundan sonraki bölümde bitirme ödevinin nasıl gerçekleştirildiğini adım adım bilgisayar yazılım mühendisliğinin gereklerine göre nasıl inşa edildiğini anlatmaya çalışacağım.

4. YAZILIM TASARIM AŞAMALARI

Bu bölümde sanal kütüphane projesinin yazılım mühendisliği gereklerine göre tasarımının nasıl yapıldığı anlatılmıştır .

4.1. Proje Amacı

4.1.1.Amaç

Bu projenin amacı, üniversitemiz bünyesinde görev yapan hocaların yine üniversite bünyesinde kendi sanal kütüphanelerini oluşturabilmelerini sağlamaktır.

4.1.2.Proje Tanımı

Sanal kütüphane projesi , kullanıcıların kendilerine ait kitapların bilgilerini girebildikleri ,kitapları ile ilgili ekleme,silme ,güncelleme gibi işlemleri yapabildikleri bir uygulamadır.

4.1.3.Proje İçeriği

Projede üniversite bünyesinde görevli hocalar sanal kütüphanenin olanaklarından faydalanabilmeleri için ilk olarak sanal kütüphaneye kullanıcı olarak kayıtlarının yaptırılmış olması gerekmektedir. Kullanıcı olarak kayıtlı olan kişilerin her birinin bir kullanıcı adı ve şifresi bulunmaktadır.Kullanıcılar ancak bu bilgilerini kullanarak, sanal kütüphane ye giriş yapabilirler.Sanal Kütüphaneye giriş yapan kullanıcılar bundan sonra genel olarak su işlemleri yapabilirler.Kütüphanede kaydı bulunan kitaplar arasında istedikleri kriterlerde arama yapabilirler, kütüphaneye kitaplarını ekleyebilir, kütüphaneden kitaplarını silebilir, kütüphanede bulunan kitaplarının bilgilerini güncelleyebilir, kullanıcı ekleyebilir veya silebilirler , istediklerinde kendi kullanıcı bilgilerini değiştirebilirler.

4.1.4.Yöntemler, Yazılım Araçları ve Platform

Yazılım Araçları : Delphi programlama Dili ve SQL sorgu dili
Platform : Windows 98 veya Windows2000 işletim sistemi

4.2. Analiz

Bu raporda, detaylı proje tanımı ve isterler analizinin verilmesi amaçlanmıştır.

4.2.1.Proje Tanımı

Bu proje; internet üzerinde sanal bir kütüphane ve bu kütüphane ile yapılan işlemleri gerçekleştirecek olan bir projedir.
Proje 6 altprogramdan oluşur.(Şekil 4.1)

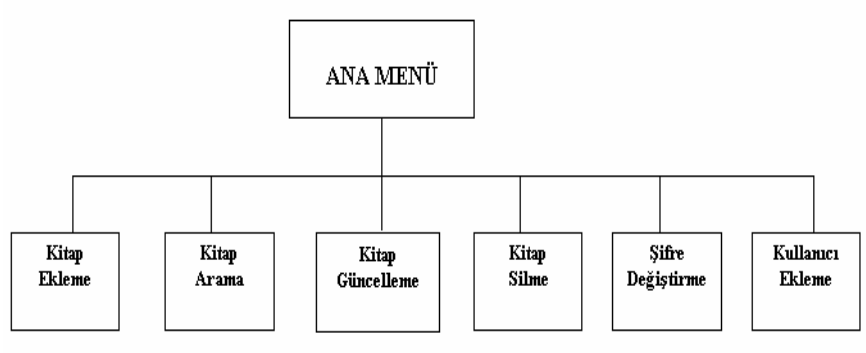
Altprogramların detaylı açıklamaları gereksinim Analizi kısmında yapılmıştır.

4.2.2.Sistem Mimarisi

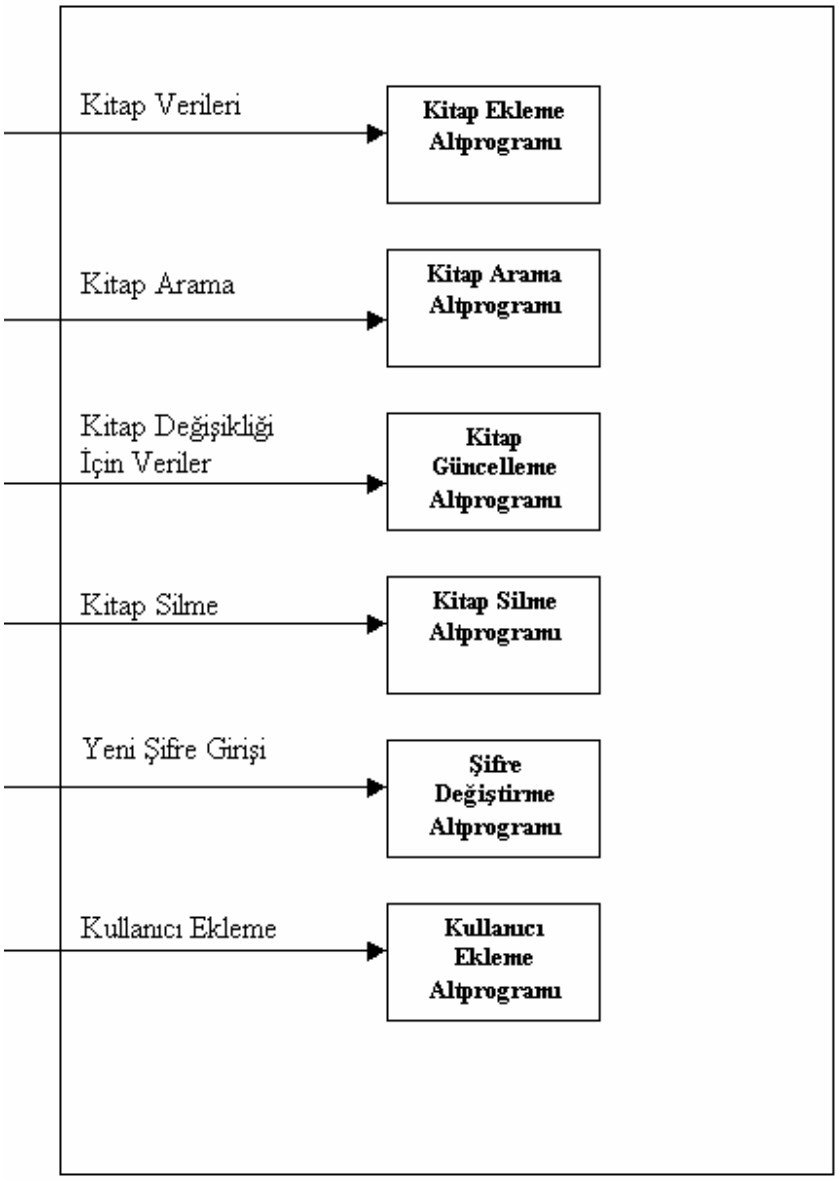
Eklenen kitaplar ve bilgileri veri dosyasında kaydedilir. Kitap Arama işleminde kütüphaneye kayıtlı kitaplar üzerinde arama işlemi yapılır Kitap değişikliği ve kitap silme işleminde kullanıcının yetkisinde olan kitaplar üzerinde işlemler yapılır. Şifre değişiminde kullanıcı yeni bir şifreye sahip olur. Kullanıcı ekleme işleminde ise kütüphaneye yeni kullanıcılar eklenir.

4.3. Gereksinim Analizi

Bu projede Yapısal Analiz yöntemi kullanılacaktır. Bu doğrultuda, sistem programının açılımı olarak 1. seviye Veri Akış Diyagramı şekil-4.2 ‘de verilmiştir:



Şekil 4.1 Altprogramlar



Şekil 4.2 1.Seviye Veri Akış Diyagramı

Her altprogramın gereksinim duyduğu bazı veriler ve gerçekleştireceği işlemler şöyle açıklanabilir;

Kitap Ekleme Altprogramı: Bu altprogramda kullanıcılar veri dosyasına yeni kitaplar eklerler. Bu bilgiler eklenirken kullanıcın verileri de eklenir. Bu sayede kullanıcılar kitap hakkında yetkiye sahip olurlar.

Kitap Arama Altprogramı : Bu altprogramda kullanıcılar kütüphaneye kayıtlı kitaplar üzerinde arama işlemi yaparlar.

Kitap Güncelleme Altprogramı : Bu altprogramda her kullanıcı kendine ait kitaplar üzerinde değişiklik yapmaktadır. Bu değişiklikten sonra yeni veriler otomatikken veri dosyasına kaydedilmektedir

Kitap Silme Altprogramı : Bu altprogramda kullanıcılar kendine ait olan kitapları silebilmektedirler. Bu işlemden sonra kitap veri dosyasından otomatikman silinmektedir.

Kullanıcı Bilgileri Güncelleme : Bu altprogramda kullanıcılar şifrelerinde değişiklik yapabilmektedirler.

Kullanıcı Ekleme Altprogramı: Bu altprogramda sanal kütüphaneye yeni kullanıcılar eklenir. Kullanıcılar kendilerine ait bir kullanıcı adı ve şifreye sahip olurlar. Bu sayede kütüphane işlemlerinden yararlanabilirler. Proje Planı ve Uygulama Takvimi
Gerçekleştirilecek görevler ve sonuç olarak sunulacak belgeler aşağıda sıralanmıştır :

Ana Görev 1 **:** Analiz Raporunun sunulması
- Görev 1 : Delphi ile SQL programlamanın öğrenilmesi
- Görev 2 : Program taslağının çıkarılması, program altprogramlarının belirlenmesi
- Görev 3 : Web sayfasının görünümünün planlanması
- Görev 5 : Tasarım Raporunun hazırlanması

Ana Görev 2 : Tasarım Raporunun sunulması

Görev 6 : Yazılımın gerçekleştirimi

Görev 7 : Gerçekleştirim Raporunun hazırlanması

Ana Görev 3 : Gerçekleştirim Raporunun sunulması

Görev 8 : Sanal verilerle yazılımın testi ve hataların giderilmesi

Görev 9 : Test Raporunun hazırlanması

Görev 10 : Proje sunumunun hazırlanması

Ana Görev 4 : Test Raporunun Sunulması

4.4. Tasarım

Bu raporda, projenin tasarımının verilmesi amaçlanmıştır. Proje tasarımı; veri tasarımı, mimari tasarım, arayüz tasarımı ve prosedürel tasarım olarak dört bölümde detaylandırılmıştır.

4.4.1.Kapsam

Bu projenin hedefi; internet ortamında bir sanal kütüphane oluşturmaktır. Oluşturulan bu sanal kütüphanenin kayıtlı kullanıcıları bulunmaktadır. Bu kullanıcılar kendilerine ait kullanıcı adı ve şifreyi kullanarak kütüphane işlemlerini yapabilmektedirler. Kullanıcıların şahsi bilgileri ve kullanıcı bilgileri veri dosyalarında tutulmaktadır. Kullanıcılar kitaplarını internet ortamına aktarmakta, bu kitaplar veri dosyasında tutulmakta ve bu sayede sanal olarak kullanıcılara açık bir kütüphane oluşmaktadır.

4.4.2.Veri Tasarımı

Bu bölümde veritabanı tablolarının tasarımı ve tablolar arasındaki ilişkileri gösteren diyagramlar verilmiştir :

4.4.2.1. Veritabanı Tabloları

Kitap tablosu

Tablo 2 Kitap Tablosu

Kullanıcı Adı	CHAR(50)	NOT NULL
Kitap_adı	CHAR(50)	NOT NULL
Yazarı	CHAR(50)	NOT NULL
Basım Tarihi	İnteger	NOT NULL
Yayın evi	CHAR(50)	NOT NULL
Konusu	CHAR(100)	NOT NULL

Tablo 3 Şifre Tablosu

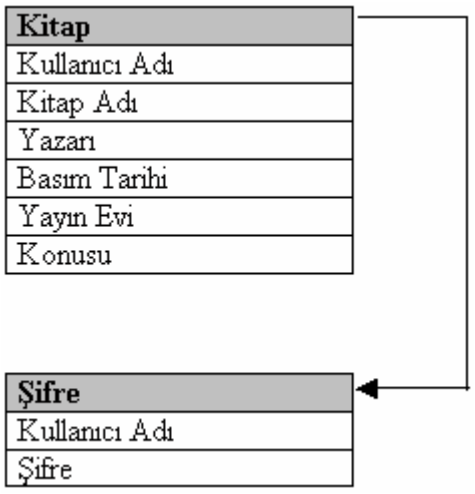
Kullanıcı Adı	CHAR(50)	NOT NULL
Şifre	CHAR(50)	NOT NULL

Primary Key (Kullanıcı Adı).

4.4.2.2. Veritabanı Diyagramı

Örnek veritabanı diyagramı :

Tablo 4 Örnek Veritabanı Diyagramı



4.4.3.Mimari Tasarım

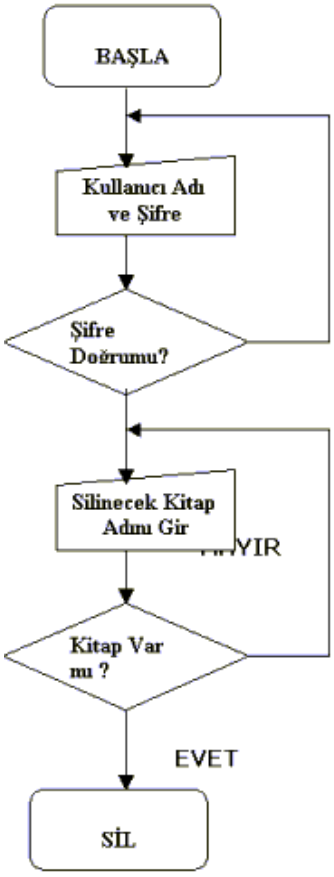
Bu bölümde sistemin modülleri arasındaki hierarşik ilişki ve altprogramlar verilmiştir.Proje 6 altprogramdan oluşur.(Şekil 4.1)

4.4.3.1.Örnek Kontrol Akış Diyagramları

Kitap Ekleme Altprogramına Ait Kontrol Akış Diyagramı:

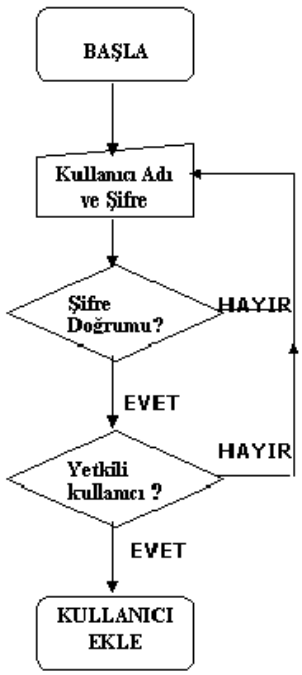
Şekil 4.3 Kitap Ekleme Altprogramına Ait Kontrol Akış Diyagramı:

Kitap Silme Altprogramına Ait Kontrol Akış Diyagramı :



Şekil 4.4 Kitap Silme altprogramına ait akış diyagramı

Kullanıcı Ekleme Altprogramına Ait Kontrol Akış Diyagramı :



Şekil 4.5 Kullanıcı ekleme altproramına ait kontrol akış diyagramı

4.5. Gerçekleştirim Raporu

Bu raporda, yazılımı oluşturan temel bileşenler ve bunların görevleri anlatılmaya çalışılmış; projenin ekran görüntülerindenkesitler verilmeye calısılmıstır.

4.5.1.Yazılımdan Kesitler:

Kütüphane Ana sayfa:

Şekil 4.6 Kütüphane Ana sayfa

Kitap Ekleme:

Şekil 4.7. Kitap Ekleme

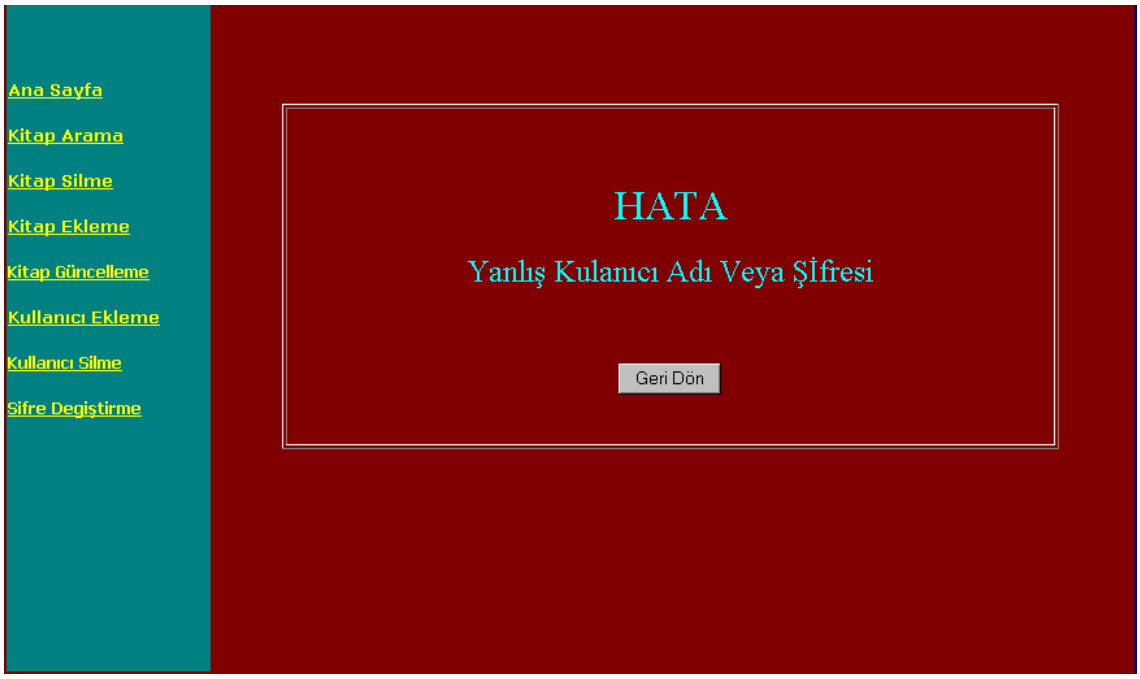
Kitap Silme :

Şekil 4.8 Kitap Silme

Kitap Arama:

Şekil 4.9 Kitap Arama

Örnek Hata Gösterimi:



Şekil 4.10 Örnek Hata Gösterimi

4.6. Test Raporu

4.6.1.Özet

Bu raporda,sanal Kütüphane projesine çeşitli test işlemleri uygulanmaya çalışılmış ve uygulamalar sonucunda elde edilen sonuçlar verilmeye çalışılmıştır.

4.6.2.Test Plani

Projenin testinde olabildiğince tüm değerlendirilmeler yapılmaya çalışılmıştır.Test sonuçları neticesinde düzeltilmesi gerekebilecek hataların düzeltilip veya hata bulma işlemine devam edilmiştir.Yazılımın test işlemi için herhangi bir test aracı kullanılmamıştır.Yazılıma yapacağımız test projenin en son halinde çeşitli örnek test verileri girilmesi seklinde olacaktır.İlk olarak yazılımın birbirinden bağımsız alt parçaları örnek verilerle test edilecek daha sonra program ünitelerinin birbirleri ile olan

ilişkileri test edilecektir.Buradaki test işlemişe ayni şekilde örnek değerlerin girilmesiyle yapılacaktır.

4.6.2.Projeyi Oluşturan Modüllerinin Bağımsız Test İşlemleri

Her modül için yeni giriş, sorgulama , güncelleme, silme işlemleri birkaç kez ve değişik veri örnekleri ile yapılmıştır.Örnek olarak girilen verilerin sonucunda , beklenen sonuçların gerçekleşip gerçekleşmediği test edilmiştir.

Kullanıcı adı ve Şifre Doğrulama :Bu modülde kütüphaneye giriş yapmak isteyen kişilerin kullanıcı adi ve şifre doğrulaması yapılır.Bumda Yanlış kullanıcı adi ve şifre değerleri girerek kontrol yaptık.Kontrol sonucunda kütüphane kullanıcısı olmadığımıza dair hata mesajı ile karsılaştık.Doğru kullanıcı adi fakat yanlış şifre değeri halinde yine hata mesajı aldık

Kitap Ekleme : Elimizde bulunan bizim belirlediğimiz bir kitabi kütüphanemize ekleyeceğiz.Kitap hakkında su veriler girilir : Kitap adi ,Kitap Yazarı ,basım yılı,basım yeri , anahtar kelimeler.Bundan sonra su kontrolleri yaptık.İlk olarak Kitaba ait bilgilerin hepsinin girilmesini kontrol ettik.Kitaba ait girilmeyen bilgi olduğu zaman mesela basım tarihi bilgisi giremediğimiz takdirde tüm verilerin girilmesi zorunluluğun olduğunu bildiren bir uyarı mesajı aldık.(istediğimiz sonuç).Başka bir kontrol eklemek istediğimiz kitabin önceden eklenmiş olması durumundadır.Buladada istediğimiz sonucu aldık.Böyle Bir kitabin olduğuna dair uyarı geldi.Basım Tarihinin sayı değeri yerine yazı ile girilmesi olasılığınıza denedik ve basım tarihi değerini yazı ile girince uyarı mesajı ile karsılaştık.

Kitap Arama :Kitap arama modülünde kullanıcı tümü,kitap adi,kitap yazarı,basım yeri ,basım yılı,anahtar kelime kriterlerinden birine göre veya birden fazla kritere göre arama yapma imkanına sahiptir.Biz burada ilk önce arama yapılırken herhangi bir bilginin girilip girilmediğini kontrol ettik.Hiçbir bilgi girmeden arama yapmayı denedik.Sonuçta Hata mesajı ile karsılaştık,yani beklediğimiz sonuçla

karsılaştık.İkinci bir kontrolde yine basım tarihi ile ilgili kontroldü.Burada da istenilen sonuç elde edilmiştir.

Kitap Bilgilerini Güncellenme :Bu modülde veritabanımızda bulunan kitabımızla ilgili bilgileri güncelleyebiliyoruz.Hangi kitabi seçersek o kitapla ilgili güncelleme olanağına sahibiz.Bu modülümüzle ilgili Su kontrolleri yapmaya çalıştık.İlk olarak hiçbir kitabi seçmeden değiştirme işlemi yapmaya çalıştık.Burada herhangi bir kitabı seçmemiz gerektiğini belirten bir mesajla karşılaştık.Bu istenilen sonuçtu.Daha sonra seçtiğimiz kitabın bilgilerini değiştirme işlemleri yaptık.Burada da kitap ekleme modülünde yaptığımız kontrollerin aynisini yaptık ve bu testlerden de basarili sonuçlar aldık.

Kitap Silme :Kitap Silme modülünde kullanıcı kütüphanemizde bulunan kendisine ait herhangi bir kitabi silebilmektedir.Bunun için ekrana gelen kitaplardan birini seçmek zorundadır.Burada kitap güncelleme modülüne uyguladığımız gibi herhangi bir seçim yapmadan silme işlemi gerçekleştirmek istedik bu durumda istediğimiz hata mesajını aldık ve test olumlu olarak sonuçlandı.

Şifre Değiştirme :Sanal Kütüphane kullanıcısı olanlar programın bu parçasıyla istediklerinde kendi kütüphane şifrelerini değiştirebilmektedir.Burada su kontrolleri yaptık.Şifre değiştirme esnasında yeni şifrenin girilmesi iki defa istenilmektedir.Farklı iki şifre değeri girdik.Bu durumda hata yaptığımıza dair uyarı aldık.Buda istediğimiz sonuçtu.

Üyeliğe Son Verme : Kütüphane programın da kullanıcılar istediklerinde kendi üyeliklerine son verebilmektedirler.Programda üyeliğe son verme tuşuna basıldığında üyeliğe son vermek istenilip istenilmediğine dair soru sorulmaktadır.Kullanıcı olumlu yanıtlarsa üyelik sona erdirilmekteki.Burada söyle bir kontrol yapabildik.Üyeliğine son verdiğimiz kişinin kullanıcı adıyla tekrar kütüphaneye girmeye çalıştık.Program bu esnada Yanlış kullanıcı hata asi vererek kütüphaneye girmemize izin vermedi.Bu istenilen sonuçtu.

4.7. Test Sonuçları

Yukarıda anlatıldığı gibi sistem ve sistemin tüm modülleri test edilmiştir.Yaptığımız kontrollerde beklediğimiz sonuçların tümüne ulaştık yani sonuçlara %100 oranında ulaşılmış.

KAYNAKLAR

CANTU , Marco., 1999, Delphi4 Uygulama Geliştirme Kılavuzu. Alfa Yayınevi.

BARENGİ , Ruhver.,1999,Delphi4'e Bakış.Seçkin Yayınevi.

YARIMAĞAN,Ünal.,2000,Veri Tabanı Sistemleri,tbv Yayınevi.

DESAİ,Bipin C.,1986,An Introduction to Database System,Web publishing company.

KALIPSIZ,Oya.,1992,Bilgisayar Yazılım Mühendisliği,İ.Ü yayınevi