VERİ DEĞİŞİMİ SAĞLAYAN BANKACILIK İŞARETMELE DİLİ: BID

Zeynep ALTAN
İstanbul Üniversitesi, Mühendislik Fakültesi,
Bilgisayar Mühendisliği Bölümü,
34850 Avcılar, İstanbul
e-posta: zaltan@istanbul.edu.tr

Anahtar Sözcükler: İşaretleme Dili, XML, veri transferi ,Visual Basic, finans sektörü

ABSTRACT

Since XML presents an independent platform for data exchange from the system and its application, banking sector is one of the most appropriate fields. The communication between the branches of any bank, and banks like EFT procedure is their fundamental operation. But the implementation of data exchanging between different banks may be complicated because of the inconsistent database systems of them. Therefore XML presents the potential solution being a suitable format. Once XML has fully been established, it is possible to extend its capabilities. This paper presents an application program supported by BID (Banking Marking Language) that performs different banking operations. XML as a meta-language describing other languages realized data relationships in the tool. Since the XML documents can easily and speedily be parsed in Visual Basic environment, we developed the tool by using this language.

1.GİRİŞ

XML, HTML'in sabit-andaç kümesine esir olmuş bilgilerin özgürlüğünü sağlayan temelinden itibaren HTML'den farklı bir teknolojidir ve en önemlisi de daha-hızlı bir HTML değildir. XML, HTML gibi bir yazılım programı değildir; bu yüzden yardımsız hiçbir şey yapamaz. Genişletilebilme özelliğine sahip XML dokümanlarında kullanılabilen elemanların sabit olmamasına rağmen, tanımlanan kuralların iyi şekillendirilmiş ve geçerli olmaları önemlidir. Bu özelliği ile, HTML'deki gibi bazı kısıtlanmış veya önceden tanımlanmış olan kelimeleri kullanma zorunluluğu ortadan kalkmıştır. Böylece, tasarımcının kendi işaretleme dilini tanımlamasının en önemli avantajı başkalarının kullandığı biçime bağlı kalmak zorunda olmayıp, bilgi ve datayı kendisinin yapılandırabilmesi ve böylece daha iyi değerlendirebilmesidir.

Genelde XML dokümanları kelimelerden oluşan *veri içeriği*, dokümanın tipi ve elemanların organizasyonunu betimleyen *yapı* ve bilginin kağıt üzerinde, tarayıcı ekranında ya da sesli olarak sunumunu gerçekleştiren *sunum* bileşenlerinden oluşur. XML'in temel fikri, sistemin bu üç bileşenden birbirlerinden bağımsız olarak yararlanmasını sağlamaktır.

1986 yılından beri XML'in yaptığından daha fazlasını gerçekleştiren uluslararası bir standart SGML(Standart Generalized Markup Language) idi. Fakat bu standart pek çok karmaşıklığı beraberinde getirmiştir. Aslında her XML dokümanı bir SGML dokümanı olarak SGML'in bir alt sınıfıdır ve SGML'i Web üzerinde kullandırmayı hedefler. Fakat SGML dokümanları birer XML dokümanı olmak zorunda değildir.

XML, HTML ve Web'in önderliğini yaptığı bilgi gösteriminin genelleştirilmiş düzenleşimi olarak görülebilir. XML, işaretleme dili yaratmak için kullanılan bir ortam iken; HTML, işaretleme dilinin örneğidir. İnternet'in kullanım alanları geliştikçe, HTML'in basitliği yerini güçsüzlüğe bırakmıştır. Web'i elektronik ticaret, sağlık bilgileri, çevrimiçi oylama gibi alanlara genişletmek için daha geniş kullanımlı, sağlam ve resmi olarak tanımlanmış standartlar gerekmiştir. Microsoft çevrimiçi bankacılık alanında, Netscape "Push" teknolojisi ile, Sun Microsystems Web otomasyonunda, Adobe veri tabanı, IBM firması ise yazılım dağıtımında XML'den yararlanmakta olan kuruluşlardan sadece birkaçıdır. XML veri değişimi için sistemden ve uygulamadan bağımsız bir platform sunduğu için, kullanıldığı en önemli alanlardan biri bankacılık sektörüdür. Bilindiği gibi bankacılıkta tüm bankaların belli yollarla birbirleriyle iletişim kurmaları, böylelikle EFT gibi iki farklı banka arasındaki işlemleri gerçekleştirebilmeleri esastır. Fakat herhangi iki farklı banka arasında veri değişimini gerçekleştirmek o kadar kolay olmayacaktır. Bunun altında yatan temel neden iki bankanın sistemlerinin ve veri tabanlarının farklı olabilmesidir. Platformdan ve sistemden farklı olarak işleyebilen bir teknoloji olan XML'in önemi bu noktada ortaya çıkar [1,2,3].

2.BID PROGRAMI

Temel bankacılık işlemlerini gerçekleştirmek üzere geliştirilen BID paket programı Visual Basic programlama dili kullanılarak geliştirilmiştir.

2.1. BID İşaretleme Dilinin Oluşturulması

Uygulama programında bankalar arasıdaki veri değişimi için XML kullanılarak, BID (Bankacılık İşaretleme Dili) adı verilen bir dil geliştirilmiş ve pakete bu isim verilmiştir. Bu dil aracılığıyla gerek bankalar arasındaki, gerekse bir bankanın şubeleri arasındaki çeşitli ödeme işlemleri gerçekleştirilmiştir.

Paket programda kullanılacak olan BID işaretleme dilinde öncelikle:

```
<!ELEMENT BID          (BANKA,ISLEM)>
<!ELEMENT  BANKA       (ADI,SUBE)>
<!ELEMENT  ADI         (#PCDATA)>
<!ELEMENT  SUBE        (#PCDATA)>
<!ELEMENT İSLEM (İSLEM_KODU,GONDEREN,ALAN?,
                 KACINCI?,MIKTAR? , TARIH,ONAY)>
.....
.....
```

şeklinde doküman tipi tanımlamalarının (DTD) ve

```
.....
.....
<complexType name="Bid_t">
   <sequence>
      <element name="BANKA"  type="Banka_t"
                  minOccurs="1" maxOccurs="1" />
      <element name="ISLEM   type="Islem_t"
                  minOccurs="1" maxOccurs="1" />
   </sequence>
</complexType>

<element name="BID" type="Bid_t" />
```

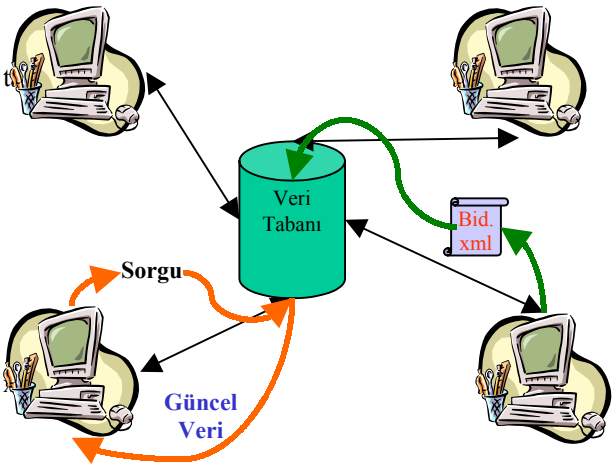
şeklinde *XMLSchema* yapılarının tanımlanmış olması gerekmektedir.

DTD ve *XMLSchema* yapısına uygun geçerli bir BID dosyası şöyle oluşturulabilir:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BID (View Source for full doctype...)>
<BID>
  <BANKA>
     <ADI>A</ADI>
     <SUBE>Avcılar</SUBE>
  </BANKA>
  <ISLEM>
    ISLEM_KODU>EFT</ISLEM_KODU>
  <GONDEREN>16980020</GONDEREN>
     <ALAN>B-16980048</ALAN>
     <MIKTAR>50000000</MIKTAR>
     <TARIH>01.06.2002</TARIH>
     <ONAY>1</ONAY>
  </ISLEM>
</BID>
```

Örnekten de görülebileceği gibi A Bankasındaki 16980020 hesap numaralı müşteri, B Bankasındaki 16980048 hesap numaralı müşteriye 01.06.2002 tarihinde 500.000.000 TL. EFT ile göndermektedir.

Yukarıdaki tanımlamalar dilin kullanabileceği andaçlar ve tipleri hakkında bilgi vermekte ve böylece geliştirilen uygulama için geçerli XML dosyası oluşturulmasına olanak sağlanmaktadır. Bu çalışmada XML dosyalarının oluşturulması ve çözümlenmesi için MSXML V3.0 kullanılmıştır. Programın genel akışı iki farklı uygulamadan oluşmaktadır. Bunlardan ilkinde aynı bankanın farklı şubelerinin herhangi bir işlem gerçekleştirilirken ortak bir veritabanına erişerek işlemlerini yürüttükleri (Şekil 1), diğerinde ise farklı iki bankanın ayrı veri tabanları kullandıkları kabul edilmiştir (Şekil 2). XML dosyalarını xml biçiminde kaydedebilmek için program içinde bir dizgi olarak oluşturulan xml dokümanını, yaratılan xml dokümanı nesnesinin .loadXML özelliğine göndermek yeterlidir



Şekil 1: Aynı bankanın farklı şubeleri arasındaki veri değişimi şeması

Program içerisinde xml dosyalarını tutan *xmlim* adında ve "object" tipinde bir değişken "global" olarak tanımlanmıştır . Programa işlemlenmek üzere dahil edilen xml dosyalarını *xml_dosya* isimli bir değişken çözümlemektedir. Bu değişken *MSXML.DomDocument* tipinde tanımlanmıştır. Bir başka ifade ile, geliştirilen paket programda xml dosyalarının çözümlenmesinde DOM (Document Object Model) kullanılmıştır. Bu modelin özelliği gereği dosya hafızaya yüklenerek ağaç yapısı oluşturulmuştur. İstenilen verilere bu ağaç üzerinden ulaşılır. Paket program içindeki *xml* dokümanındaki *hesapno* değerine ulaşan bir satır aşağıdaki gibi yazılabilir:

hesapno=*xml_dosya.childNodes(2).childNodes(1).childNodes(1).nodeTypedValue.*

Paket programda bu işlem için kullanılması gereken değerlerin tümü yukarıdaki satıra benzer şekilde elde edilebilir.

2.2 Paket Programın Oluşturulması

Programın kodu Visual Basic 6.0 programlama dilinde yazıldığı için, veriler Access veritabanında tutulmaktadır. Veritabanındaki alanlar hesap numarası ile ilişkilendirilmekte, böylece hesap numarası bilinen bir müşterinin kişisel bilgilerine ulaşmak mümkün olmaktadır. Benzer şekilde hesap numarası bilinen birine ait borç bilgileri de görüntülenebilmektedir. Veritabanındaki bilginin sorgulanması SQL(Structured Query Language) kullanarak gerçekleştirilmiştir. Dönen veriler DBGrid nesnesinde gösterildiği gibi bazı değişkenlere atanarak işlemlerin devamlılığı sağlanmıştır.

Geliştirilen BID programında havale, EFT, otomatik ödeme, vezne ödemesi, müşteri ekleme, para yatırma gibi farklı işlemler gerçekleştirilmektedir.

2.2.1Havale İşlemi

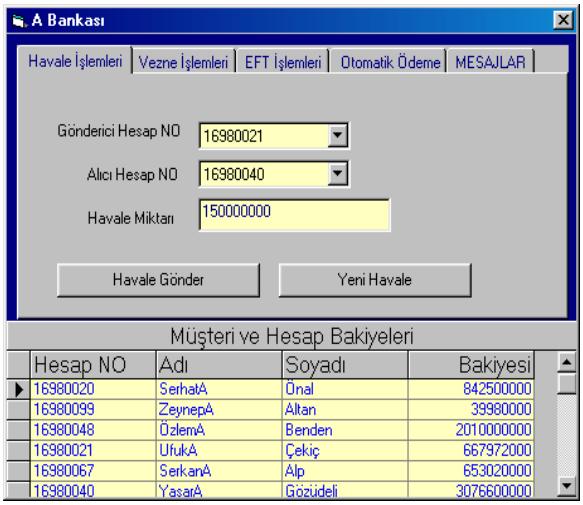
Havale işlemi aynı bankanın farklı şubeleri ya da aynı şube içerisinde gerçekleşir. Havale işleminde hem alıcının, hem de gönderenin hesabı aynı bankada, yani aynı veritabanında saklıdır. Gönderen ve alıcının hesap numaraları girildikten ve gönderilmek istenen para miktarı yazıldıktan sonra "Havale Gönder" düğmesine basılınca gönderilecek olan havale miktarının gönderenin bakiyesinden daha fazla olup olmadığına bakılır (Şekil 3). Fazla ise işlem gerçekleştirilmez. Eğer bakiyesi havale işlemi için yeterli ise düğmeye basıldığında havale işlemini gerçekleştirecek *bid.xml* dosyasını oluşturur. Bu dosya şöyle oluşturulur:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BID(View Source for fulldoctype)>
<BID>
  <BANKA>
      <ADI>A</ADI>
       <SUBE>Avcılar</SUBE>
   </BANKA>
   <ISLEM>
       <ISLEM_KODU>HAVALE
                   < /ISLEM_KODU>
         <GONDEREN>16980020
                   </GONDEREN>
       <ALAN>16980067</ALAN>
       <MIKTAR>150000000</MIKTAR>
       <TARIH>02.06.2002</TARIH>
       <ONAY>1</ONAY>
  </ISLEM>
</BID
```

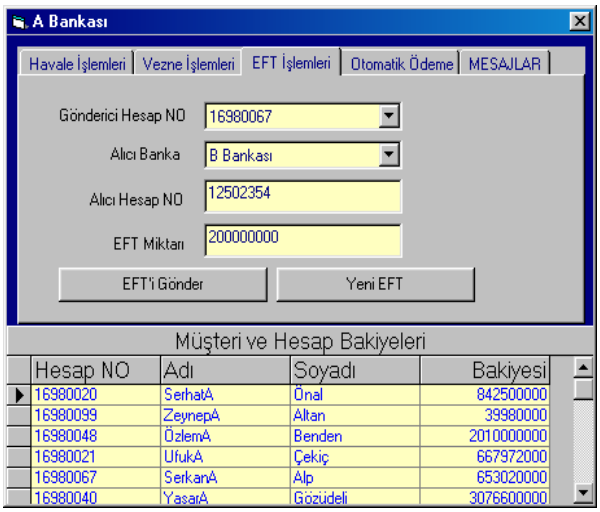
Yukarıdaki dosya uygulamaya dönüştüğünde girilen havale miktarı gönderenden düşülür ve aynı miktar alıcı müşterinin bakiyesine eklenir. Bu işlemlerden sonra programda işleme giren veriler güncellenir.

2.2.1EFT İşlemi

Farklı bankalar arasında gerçekleştirildiği için EFT daha karmaşık bir işlemdir. Bu işlemde müşteri, kendi bankasından EFT göndermek istediği kişinin hangi bankanın şubesinde hesabı olduğunu bilmeli ve hesap numarasını da belirtmelidir (Şekil 4). Çeşitli bankalarla çalışıldığı için farklı platformlar ve veri tabanları ile karşılaşılacaktır. Bu nedenle xml kullanmak artık zorunludur.



Şekil 3: Bir hesaptan başka hesaba havale işlemini gösteren arayüz



Şekil 4: Bir bankadan başka bankaya EFT işlemine ait arayüz

Bu işlemde gönderenin hesap numarası hesabının bulunduğu bankanın veri tabanında mevcuttur. Ama alıcının hesap numarası o bankada olmadığı için bunun kontrolü yapılamaz. Havalede olduğu gibi bu işlemde de EFT miktarının kullanıcı bakiyesinden az

olması gerekir. Bu kontrol de yapıldıktan sonra dosya belirtilen bankaya yollanır. İlgili dosya aşağıdaki gibi hazırlanabilir:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <!DOCTYPE BID (View Source for full
   doctype...)>
<BID>
   <BANKA>
       <ADI>A</ADI>
       <SUBE>Avcılar</SUBE>
    </BANKA>
    <ISLEM>
         <ISLEM_KODU>EFT_ONAY
                  </ISLEM_KODU>
         <GONDEREN>16980067
                  </GONDEREN>
          <ALAN>B-12502354</ALAN>
        <MIKTAR>200000000</MIKTAR>
         <TARIH>02.06.2002</TARIH>
         <ONAY>1</ONAY>
    </ISLEM>
 </BID>
```

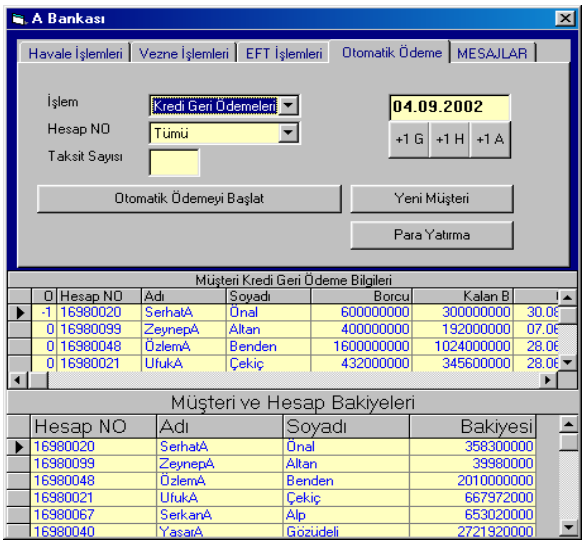
Dosya gönderildiği zaman gönderenin hesabından para hemen düşülmez. Çünkü girilen alıcı hesap numarası karşı bankada bulunmayabilir. İşte bu noktada iki farklı durum ile karşılaşılabilir. Eğer karşı tarafta kullanıcı bulunduysa belirtilen miktar alıcının bakiyesine eklenir ve karşı bankadan bir *bid.xml* dosyası daha gönderilir. Bu dosyada işlemin gerçekleştirildiğini belirten alanlar mevcuttur. İşlem kodu EFT_ONAY olan bu dosyada ONAY andacı 1 değerindedir. Bu dosya alındığında, uygulama işleminin karşı tarafta gerçekleştiğini anlar ve göndericinin bakiyesini düşer.

İkinci durum ise karşı bankada ilgili hesap numarasının bulunmaması durumudur. Bu durumda karşı banka yine EFT_ONAY işlem kodlu bir bid.xml dosyası yollar. Fakat bu sefer bu dosyadaki ONAY andacının değeri 0 olarak gelir. Bu durumda gönderenin bankasında bu işlemin gerçekleşmeyeceği anlaşılır ve işlem iptal edilir.

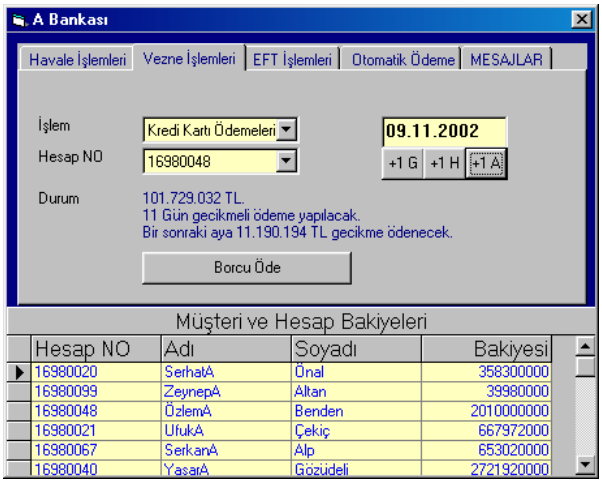
2.2.3 Otomatik Ödeme

Otomatik ödeme sistemi, bankadan alınmış olan kredi borçlarının geri ödemesi ya da kredi kartı borçlarının her ay otomatik ödenmesi olarak düzenlenmiştir. Her iki işlem için de veritabanında bir önceki ödemenin tarihi tutulmaktadır. Böylece borçlunun bir sonraki ödemesi, veritabanındaki tarih ile karşılaştırılarak gerçekleşecektir. Paket programda ödemenin yapıldığı tarihi gösteren ve bu tarihi gün, hafta ve ay olarak ilerleten düğmeler kullanılmış, böylece otomatik ödemelerin takibi kolaylaştırılmıştır (Şekil 5).

Otomatik ödemeyi başlatmak için öncelikle işlem türünün seçilmesi gerekmektedir. İşlem kredi borcu ise, ilk ödemede alınan kredinin kaç ayda ödeneceği belirlenir ve otomatik ödeme talimatı verilir. Her ay otomatik ödemenin yapılacağı günde o aya ait taksit miktarı müşterinin hesabından düşer. Eğer ödeme günü müşterinin bakiyesi taksit miktarı için yetersiz ise, borçlu para yatırdığında, havale geldiğinde ya da EFT ile para gönderildiğinde daha önce yatırılmamış olan taksitler için faizi hesaplanarak bakiyesinden düşürürülür.



Şekil 5: Otomatik ödeme talimatı veren tüm müşterilere ait borç ve hesap durumlarını gösteren arayüz



Şekil 6 :Vezneden kredi kartı ödemesine ait arayüz

Yapılacak işlem kredi kartı borcu ödemesi ise, her ay belirli günde müşterinin kredi kartı borcu bakiyesinden düşülür. Eğer bakiyede yetersiz para varsa, kredi borcu ödemelerinde olduğu gibi, hesaba para geldiğinde faizi ile birlikte kredi kartı borcu düşülür.

2.2.4. Vezneden Ödeme

Otomatik ödeme talimatı veren müşterilerin dışında kalanlar, aldıkları kredilerin geri ödemelerini ya da her ay kredi kartı borçlarını bankadan yatırırlar. Vezneden ödemelerde, kredi borcu için borç miktarı 5 takside bölünür. Eğer ödeme geç yapılırsa kredi taksiti faiziyle müşteriden istenir. İşlem eğer kredi kartı borcunu ödeme işlemi ise her ay borç tarihinde müşterilerin borçlarını ödemesi beklenir. Bu işlemde

borçlarını yatırmakta geciken müşterilerin gecikme faizleri bir sonraki aya faiz olarak aktarılır (Şekil 6).

2.2.5 Diğer İşlemler

Yeni müşteri eklemek için geliştirilen arayüzde, müşteri bilgileri kayıt edilmekte ve hesap açılmaktadır. Para yatırma arayüzü ise, müşterinin bankadan gerçekleştirdiği para yatırma işlemini görüntüler. Bu para yatırma işleminden sonra eğer ödenmemiş kredi kartı borçları ya da kredi geri ödeme taksitleri varsa, bu taksitler otomatik olarak bakiyelerinden düşülecektir.

Aşağıda BID programının havale, EFT ve otomatik kredi geri ödemesi işlemlerine ait Visual Basic 6.0'da yazılmış kod parçası görülmektedir:

```
Public xml_dosya As MSXML.DOMDocument
Public bankanin_adi As String
.....
 islem = xml_dosya.childNodes(2).childNodes(1).childNodes(0).
                                          nodeTypedValue
Select Case islem
Case "HAVALE"
miktar = Trim(xml_dosya.childNodes(2).childNodes(1).
                        childNodes(3).nodeTypedValue)
gonderen = Trim(xml_dosya.childNodes(2).childNodes(1).
                        childNodes(1) . nodeTypedValue)
alan = Trim(xml_dosya.childNodes(2).childNodes(1).
                         childNodes(2).nodeTypedValue)
sql = " update hesap set bakiye = bakiye - " + miktar + _
                  " where hesap_no = ' " + gonderen + " ' "
db.Execute (sql)
sql = "update hesap set bakiye = bakiye + " + miktar + _
                  " where hesap_no = ' " + alan + " ' "
db.Execute (sql)
 List1.AddItem Time() & " :: " & gonderen & " -> " & alan & " :"
           &  Format(miktar, "##,##") & " havale göndermiştir. "
       otomatik_kontrol
       otomatik_kontrol_kart
Case "EFT"
gelen_banka = Mid(Trim(xml_dosya.childNodes(2).childNodes(0).
                        childNodes(0).nodeTypedValue), 1, 1)
alan = Mid ( Trim (xml_dosya.childNodes(2).childNodes(1).
                                childNodes(2).nodeTypedValue), 3,
                Len (Trim (xml_dosya.childNodes(2).childNodes(1).
                         childNodes(2).nodeTypedValue)) - 2)
miktar = Trim(xml_dosya.childNodes(2).childNodes(1).
                              childNodes(3).nodeTypedValue)
gonderen = Trim(xml_dosya.childNodes(2).....)
sql = "select*from hesap  where hesap_no = ' "+ alan + " ' "
Set rs = db.OpenRecordset(sql, dbOpenDynaset)
If rs.EOF Then
onay = 0
List1.AddItem Time() & " :: " & gelen_banka & "
                bankasından gelen EFT'e onay verilmedi."
Else
sql = "update hesap set bakiye = bakiye + " + miktar + _
              " where hesap_no = ' " + alan + " ' "
db.Execute (sql)
onay = 1
List1.AddItem Time() & " :: " & gelen_banka & "
               bankasından gelen EFT'e onaylandı."
.....
rs.Close
Set xmlim = CreateObject("microsoft.xmldom")
.....
Case "OTOMATIK"
gonderen = Trim( xml_dosya.childNodes(2).....)
tarihh = Trim(xml_dosya.childNodes(2).
          childNodes(1).childNodes(3).nodeTypedValue)
kacinci = Trim(xml_dosya.childNodes(2).
          childNodes(1).childNodes(2).nodeTypedValue)
Set db = OpenDatabase(App.Path & "\db1.mdb")
sql ="select b.borcu, h.bakiye from borclar b, hesap h
        where b.hesapno= ' " & gonderen & " ' and
                                  b.hesapno=h.hesap_no
Set rs = db.OpenRecordset(sql)
bakiyesi = CCur(rs.Fields(1).Value)
taksidi = CCur(CCur(rs.Fields(0).Value) / kacinci)
rs.Close
List1.AddItem Time() & " :: " & gonderen & " hesap no'su
            " & kacinci & " taksitli otomatik ödeme açtı."
If bakiyesi >= taksidi Then
 sql ="update borclar set odeme_bas=' " & tarihh & "  ',
            taksit_sayisi= " & kacinci & ", kalan_taksit_sayisi=
    " & kacinci –1  & ", otomatik=1, kalan_borc=borcu-borcu/
      " & kacinci & "  where  hesapno='" & gonderen &" ' "
 db.Execute (sql)
 sql = "select borcu-kalan_borc as x from borclar
                  where hesapno= ' " & gonderen & " ' "  Set rs =
db.OpenRecordset(sql)
nekadar = CCur(rs.Fields( "x" ))
Set rs = Nothing
sql = "update hesap set bakiye=bakiye- " & nekadar & "
        where hesap_no= ' "  & gonderen & " ' "
db.Execute (sql)
List1.AddItem Time() & " :: " & gonderen & " den " & kacinci & "
                        otomatik ödeme düşüldü."
Else
sql = "update borclar set odeme_bas=' " & CStr(DateAdd( "m" , -1,
                    tarihh))  & " '  , taksit_sayisi="  & kacinci & ",
                kalan_taksit_sayisi = " & kacinci & ", otomatik=1,
          kalan_borc=borcu    where hesapno=' " & gonderen & " ' "
db.Execute (sql)
List1.AddItem Time()  & " :: " & gonderen & " in bakiyesi yetersiz.
                    " Ödemesi düşülemedi. "
 End If
 borc_update " ", 1
 otomatik_listesi
 otomatik_olmayan_listesi 1
call Combo4_Click
```

3. SONUÇ

Bu yazıda günümüzde özellikle finans sektöründe oldukça güncel olan XML teknolojisi kullanılarak geliştirilen bankacılık programı anlatılmaktadır. Bankacılık programını oluştururken öte bir dil olan XML ile yaratılmış Bankacılık İşaretleme Dili *BID* kullanılmış, programdaki veri alışverişleri bu dil üzerinden gerçekleştirilmiştir. Aynı isim verilen paket program geliştirilirken gerek hızı ve gerekse etkin programlama özellikleri nedeni ile Visual Basic 6.0 programlama dili tercih edilmiştir. Pek çok yazılım evlerinde de benzeri programların tasarımı için tercih edilen Visual Basic programlama dilinin, XML dosyalarını oldukça kolay ve hızlı şekilde çözümleyebilme özelliği vardır.

Teşekkür:
Çalışmaya katkılarından dolayı öğrencim Serhat Önal'a teşekkür ederim.

KAYNAKÇA

[1] Tim Bray ,"Extensible Markup Language 1.0 (2. Edit.)", http:/www.w3.org/TR/REC-xml,2000.

[2] Altova (Download Center) , "XML Spy 4.2 Suite Setup", http:/new.xmlspy.com/download.html

[3] Peter Flynn (editor) , "The XML FAQ", http:/www.ucc.ie/xml, 2002.