İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇEVRİMİÇİ ÇALIŞAN BİR KREDİ KARTI İŞLEM SİSTEMİ

YÜKSEK LİSANS TEZİ

Müh. Özgür TABAK

Anabilim Dalı : KONTROL VE BİLGİSAYAR MÜHENDİSLİĞİ

Programı : KONTROL VE BİLGİSAYAR MÜHENDİSLİĞİ

ARALIK 2002

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇEVRİMİÇİ ÇALIŞAN BİR KREDİ KARTI İŞLEM SİSTEMİ

YÜKSEK LİSANS TEZİ
Müh. Özgür TABAK
504981156

Tezin Enstitüye Verildiği Tarih : 24 Aralık 2002
Tezin Savunulduğu Tarih : 16 Ocak 2003

Tez Danışmanı : **Prof.Dr. Bülent ÖRENCİK**
Diğer Jüri Üyeleri **Prof.Dr. M. Yahya KARSLIGİL (Y.T.Ü.)**
Doç.Dr. Coşkun SÖNMEZ

ARALIK 2002

ÖNSÖZ

Kredi kartı, finans endüstrisinin insanlara sunduğu en önemli ve kullanışlı ürünlerden birisidir. Bilgisayar ve elektronik teknolojisindeki gelişmeler sayesinde kredi kartlarının kullanım alanları ve yaygınlığı her geçen gün artmaktadır. Günümüzde kredi kartları ile para çekme makinaları, satış noktaları, Internet, telefon gibi çeşitli kaynaklardan işlem yapılabilmektedir. İlgili kaynaklardan yapılan işlemler kartın ait olduğu finansal kuruma yönlendirilmekte ve finansal kurumun yaptığı kontrollere göre işlemin onaylanıp onaylanmayacağına karar verilmektedir.

Bu çalışmada çevrimiçi olarak çalışan bir kredi kartı işlem sistemi tasarlanmıştır. Tez çalışmam süresince yardımlarından dolayı danışman hocam Sayın Prof.Dr. Bülent Örencik'e ve her konuda bana destek olan aileme teşekkür ederim.

Aralık, 2002 Özgür TABAK

İÇİNDEKİLER

KISALTMALAR **v**
TABLO LİSTESİ **vi**
ŞEKİL LİSTESİ **vii**
ÖZET **ix**
SUMMARY **x**

1. GİRİŞ **1**

2. KAVRAMLAR VE KULLANILAN TEKNOLOJİLER **3**
2.1. Kavramlar 3
2.2. Kullanılan Teknolojiler 4

3. ISO8583 MESAJ FORMATI **5**
3.1. Mesaj Formatının Amacı 5
3.2. Mesaj Formatının Bileşenleri 5

4. TCP/IP PROTOKOLLERİ **8**
4.1. TCP/IP Yaklaşımı 8
4.2. TCP ve IP'nin Çalışması 8

5. INDY **10**
5.1. Indy Teknolojisi 10
5.2. Indy Paketinde İstemci, Sunucu Mimarisinin Gerçeklenmesi 10
5.3. Engelleyici Soket Yapısı 11
5.3.1. Engelleyici soketler 11
5.3.2. Engelleyici soketlerin avantajları 12
5.3.3. Engelleyici soketlerin dezavantajları 13
5.4. TIdAntiFreeze Bileşeni 13
5.5. Çok Görevcikli Yapı 13
5.5.1. Çok görevcikliliğin avantajları 13
5.5.2. Görevcik havuzlama 14
5.5.3. Görevcikler ve global kısım 15
5.6. Indy Metodolojisi 15

6. ADO VERİ ERİŞİM YÖNTEMİ **18**
6.1. ODBC, OLE DB ve ADO 18
6.2. ADO Mimarisi 19
6.3. ADO Programlama Modeli 20

7. SSL GÜVENLİK MEKANİZMALARI **22**
7.1. SSL ile Kullanılan Şifrelemeler 24
7.2. SSL El Sıkışma 25
7.3. Sunucu Doğrulama 27

7.4. İstemci Doğrulama 29

8. SİSTEM BİLEŞENLERİ VE ÇALIŞMA MANTIĞI 33
8.1. Simülatör Ekranları 34
8.1.1. Kaynak tanımlama ekranları 35
8.1.1.1. Finansal kurum tanım ekranı 36
8.1.1.2. Üye işyeri tanım ekranı 36
8.1.1.3. Cihaz tanım ekranı 37
8.1.2. İşlem simülasyon ekranları 38
8.1.2.1. Kaynak simülasyonu 39
8.1.2.2. Mesaj simülasyonu 40
8.2. İşlem Sunucusu 42
8.2.1. İşlem sunucusunun başlatılması 42
8.2.2. İşlem sunucusunun menüleri 43
8.2.3. İşlem sunucusunun çalışma mantığı 44
8.2.4. Veri tabanı bağlantılarının dinamik olarak ayarlanması 45
8.3. Kredi Kartı Veri Tabanı ve Üzerinde Çalışan Programlar 46
8.3.1. İşlem isteği mesajları 46
8.3.2. Tavsiye mesajları 47
8.3.3. İptal mesajları 48
8.4. Sorgulama ve Güncelleme Ekranları 49
8.4.1. Sorgulama ekranları 49
8.4.1.1. Kart bilgileri ekranı 49
8.4.1.2. Kartlar ekranı 49
8.4.1.3. Kart arama ekranı 50
8.4.1.4. Grup bilgisi ekranı 51
8.4.1.5. Kart hareketleri ekranı 51
8.4.1.6. Kullanılabilir limitin hareketlere göre değişimi ekranı 52
8.4.1.7. Grup kullanılabilir limitinin hareketlere göre değişimi ekranı 52
8.4.1.8. Yanıt kodları ekranı 53
8.4.1.9. Kart durum kodları ekranı 53
8.4.2. Güncelleme ekranları 54
8.4.2.1. Döviz çevirme oranları ekranı 54

9. PERFORMANS ANALİZİ 55
9.1. İstemci Yönetim Programı 56
9.2. İstemci Program 57
9.3. İşlem Sunucusu ile İlgili Ölçüm ve Grafikler 59
9.3.1. Veri tabanı bağlantı sayısı sabit 61
9.3.2. Veri tabanı bağlantı sayısı dinamik 64
9.3.3. Veri tabanı bağlantılarının her istek için açılıp, kapatıldığı durum 70
9.3.4. Veri tabanı bağlantı kurma yöntemlerinin karşılaştırılması 71

10. SONUÇLAR VE TARTIŞMA 73

KAYNAKLAR 75

ÖZGEÇMİŞ 76

KISALTMALAR

ATM	: Automatic Teller Machine
POS	: Point of Sale
ISO	: International Standards Organization
INDY	: Internet Direct
TCP	: Transmission Control Protocol
IP	: Internet Protocol
ODBC	: Open Database Connectivity
OLE DB	: Object Linking and Embedding Database
ADO	: ActiveX Data Objects
DAO	: Data Access Objects
RDO	: Remote Data Objects
SSL	: Secure Sockets Layer
HTTP	: HyperText Transport Protocol
LDAP	: Lightweight Directory Access Protocol
IMAP	: Internet Messaging Access Protocol
CA	: Certificate Authority
DES	: Data Encryption Standard
ABD	: Amerika Birleşik Devletleri
DSA	: Digital Signature Algorithm
KEA	: Key Exchange Algorithm
MD5	: Message Digest 5
RC	: Rivest Encryption Ciphers
RSA	: Rivest, Shamir and Adleman
SHA-1	: Secure Hash Algorithm 1
BVBS	: Başlangıç Veri Tabanı Bağlantı Sayısı
MAKY	: Maksimum Kullanım Yüzdesi
MNKY	: Minimum Kullanım Yüzdesi
BDS	: Bağlantı Değişim Sayısı

TABLO LİSTESİ

Sayfa No

Tablo 3.1 ISO8583 mesaj tipi ve açıklamaları……..6

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1 : Kredi kartı işlem akışı 4
Şekil 3.1 : Mesaj formatı 5
Şekil 5.1 : Indy sunucusunun çalışma yöntemi 16
Şekil 6.1 : ADO mimarisi 19
Şekil 6.2 : ADO programlama modeli 20
Şekil 7.1 : SSL'in TCP/IP ve uygulama protokolleri ile ilişkisi 22
Şekil 7.2 : İstemcinin sunucu sertifikasını doğrulaması 28
Şekil 7.3 : Sunucunun istemci sertifikasını doğrulaması 30
Şekil 8.1 : Sistem bileşenleri 33
Şekil 8.2 : Kredi Kartı İşlem Sistemi Arayüz Programı 34
Şekil 8.3 : Simülatör ekranları menüsü 34
Şekil 8.4 : Kaynak tanımlama ekranları menüsü 35
Şekil 8.5 : Tanımlama ekranlarının bölümleri 35
Şekil 8.6 : Finansal Kurum Tanım Ekranı 36
Şekil 8.7 : Üye İşyeri Tanım Ekranı 37
Şekil 8.8 : Cihaz Tanım Ekranı 37
Şekil 8.9 : İşlem simülasyon ekranlarının çalışma mantığı 38
Şekil 8.10 : İşlem Simülasyon Ekranları Menüsü 38
Şekil 8.11 : K aynak Liste Ekranı 39
Şekil 8.12 : İşlem Simülasyon Ekranı 39
Şekil 8.13 : İşlemin onaylanması durumunda verilen uyarı mesajı 40
Şekil 8.14 : İşlemin onaylanamaması durumunda verilen uyarı mesajı 40
Şekil 8.15 : Mesaj Simülasyon Ekranı – İstek Mesajı 41
Şekil 8.16 : Mesaj Simülasyon Ekranı – Yanıt Mesajı 42
Şekil 8.17 : İşlem sunucusu başlatılma mantığı 42
Şekil 8.18 : İşlem sunucusunun başlangıç durumu 43
Şekil 8.19 : İşlem Sunucusu Dosya Menüsü 43
Şekil 8.20 : İşlem Sunucusu Ayarlar Menüsü 43
Şekil 8.21 : İşlem sunucusunun çalışması 44
Şekil 8.22 : İşlem sunucusu çalışma mantığı 45
Şekil 8.23 : Kart Bilgileri Ekranı 49
Şekil 8.24 : Kartlar Ekranı 50
Şekil 8.25 : Kart Arama Ekranı 50
Şekil 8.26 : Grup Bilgisi Ekranı 51
Şekil 8.27 : Kart Hareketleri Ekranı 51
Şekil 8.28 : Kullanılabilir Limitin Hareketlere Göre Değişimi Ekranı 52
Şekil 8.29 : Grup Kullanılabilir Limitinin Hareketlere Göre Değişimi Ekranı 52
Şekil 8.30 : Yanıt Kodları Ekranı 53
Şekil 8.31 : Kart Durum Kodları Ekranı 53
Şekil 8.32 : Döviz Çevirme Oranları Ekranı 54
Şekil 9.1 : Ölçüm ve Grafiklerin Elde Edilmesinde Kullanılan Programlar 55
Şekil 9.2 : İstemci Yönetim Programı 56

Şekil 9.3 : İstemci yönetim programı çalışma mantığı 57
Şekil 9.4 : İstemci program çalışma mantığı 58
Şekil 9.5 : İstemci Sayısı – Zaman Grafiği 59
Şekil 9.6 : Yanıt Süresi – Zaman Grafiği 61
Şekil 9.7 : İşlem Sayısı – Zaman Grafiği 61
Şekil 9.8 : Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği 62
Şekil 9.9 : Yanıt Süresi – Zaman Grafiği 62
Şekil 9.10 : İşlem Sayısı – Zaman Grafiği 63
Şekil 9.11 : Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği 63
Şekil 9.12 : Yanıt Süresi – Zaman Grafiği 65
Şekil 9.13 : İşlem Sayısı – Zaman Grafiği 65
Şekil 9.14 : Açılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği 66
Şekil 9.15 : Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği 66
Şekil 9.16 : Yanıt Süresi – Zaman Grafiği 67
Şekil 9.17 : İşlem Sayısı – Zaman Grafiği 68
Şekil 9.18 : Açılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği 68
Şekil 9.19 : Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği 69
Şekil 9.20 : Yanıt Süresi – Zaman Grafiği 70
Şekil 9.21 : İşlem Sayısı – Zaman Grafiği 71

ÇEVRİMİÇİ ÇALIŞAN BİR KREDİ KARTI İŞLEM SİSTEMİ

ÖZET

Kredi kartları ile yapılan işlemlerde kartı üreten ve finansal olarak işleten kurumdan onay almak gerekir. Bu çalışmada, çeşitli kaynaklardan gönderilen kredi kartı işlem isteklerini karşılayan ve işlemin türüne göre gerekli finansal kontrolleri yaparak, işlemin onaylanıp onaylanmayacağına karar veren bir sistem tasarlanmıştır.

Sistem, simülatör programı, işlem sunucusu, kredi kartı veri tabanı, sorgulama ve güncelleme ekranlarından oluşmaktadır. Kredi kartı işlem kaynaklarını fiziksel olarak temin etmek mümkün olmadığından bu kaynaklar simülatör programı aracılığıyla tanımlanmakta ve simüle edilmektedir. Simülasyon ekranlarından işlem kaynağı ve işlem tipi seçimi yapılır, işlem için gereken diğer bilgiler girilir ve bu bilgiler kullanılarak ISO8583 formatında bir istek mesajı oluşturulur. Simülasyon programı işlem sunucusu ile TCP/IP bağlantısı kurarak oluşturduğu istek mesajını işlem sunucusuna gönderir. İşlem sunucusu kendisine gönderilen her istek mesajı için veri tabanındaki bir programı çalıştırır. Veri tabanındaki program işlem ile ilgili finansal kontrolleri yaparak, işlemin onaylanıp onaylanmayacağına karar verir ve işlem sonucu ile ilgili bilgi döndürür. İşlem sunucusu, veri tabanındaki programın döndürdüğü bilgileri kullanarak ISO8583 formatında bir yanıt mesajı oluşturur ve bu yanıt mesajını simülatör programına gönderir. Simülatör programı aldığı yanıt mesajını değerlendirir ve işlem sonucunu ekranda gösterir.

Çeşitli kaynaklardan yapılan bütün finansal işlemlerde olduğu gibi kredi kartlarında da işlem güvenliği oldukça önemlidir. Kredi kartlarında sağlanan güvenlik mekanizmalarının bir kısmı kuruluşa veya uygulamaya özgüdür, bunların yanı sıra güvenlik ve işlem akışı konusunda uluslararası kuruluşlar tarafından geliştirilmiş birtakım standartlar da vardır. Bu çalışmada, simülatör ve işlem sunucusu arasındaki mesajların güvenliği, SSL yöntemi ile şifreleme yapılarak sağlanmıştır.

Ayrıca, bu çalışmada işlem sunucusu üzerinde performans analizleri yapılmıştır. İşlem sunucusunun veri tabanına bağlantı kurma yöntemleri değiştirilerek farklı yük koşulları altında ölçümler yapılmış ve elde edilen ölçümler kullanılarak çeşitli grafikler oluşturulmuştur. Elde edilen grafiklere dayanarak işlem sunucusunun veri tabanı bağlantı kurma yöntemleri karşılaştırılmıştır.

ONLINE CREDIT CARD SYSTEM

SUMMARY

In order to perform a credit card transaction, it is mandatory to get approval from the issuer institution of the card. In this study, a system is designed in order to meet the credit card transaction requests sent from the various resources and in order to determine whether the transaction will be authorized by making the financial controls according to the transaction type.

The components of the system are the simulator program, transaction server, credit card database, inquiry and update screens. Credit card transaction sources are defined and simulated by the simulator program because of the difficulty in providing the actual sources. Transaction type, source and other information are entered from the simulator screen and an ISO8583-formatted transaction request is constructed by using these information. Simulator sends the request by setting up a TCP/IP connection with the transaction server. Transaction server executes a database program for every incoming request. The database program determines whether the transaction will be authorized by making the required financial controls and returns the result of the request. The transaction server constructs an ISO8583-formatted response message by using the results of the database program and sends the response to the simulator program. The simulator program evaluates the response message and displays it on the screen.

Transaction security is very important for the credit card transactions as for the financial transactions originated from other sources. Some of the security mechanisms which are provided for the credit cards are specific to the institution or application. In addition, there are also some international standards for the security and transaction flow. In this study, the security of the messages which are sent between the simulator and transaction server is provided by the SSL cryptography method.

In addition, in this study, performance analysis is done on the transaction server. By changing the database connection setup method of the transaction server, various measurements are made under the different load conditions and several graphics are created by using these measurements. Database connection setup methods of the transaction server are compared depending on these graphics.

1. GİRİŞ

Kredi kartı, günümüz finans endüstrisinin insanlara sunduğu en önemli kolaylıklardan biridir. Kredi kartlarının ortaya çıkması insanların nakit paraya olan ihtiyaçlarını azaltmıştır. Kredi kartları ile yapılan alışverişlerde harcanan tutarın hemen ödenme zorunluluğu olmaması da bir avantajdır.

Bilgisayar ve elektronik teknolojisindeki gelişmeler, kredi kartlarının kullanım alanlarını ve yaygınlığını arttırmaktadır. Günümüzde kredi kartları, para çekme makinaları, satış noktaları, Internet, telefon gibi çeşitli kaynaklardan harcama, nakit para çekme ve borç ödeme gibi temel işlemlerde kullanılabilmektedir.

Kredi kartları ile yapılan işlemlerde kartı üreten ve finansal olarak işleten kurumdan onay almak gerekir. İşlem kaynağı kartın ait olduğu kurum ile bağlantı kuramazsa ve işlem tutarı belirlenmiş limitlerin altında ise kuruma sormadan da işleme onay verilebilmektedir. Fakat, kart sahibini güvence altına almak ve işlem güvenliğini sağlamak açısından genelde izlenen yöntem, işlemin çeşitli kontrollerden geçirilerek onaylanıp onaylanmayacağına karar verilmesidir.

Bu çalışmada çevrim içi çalışan bir kredi kartı işlem sistemi gerçeklenmiştir. Sistem, simülatör programı, işlem sunucusu, kredi kartı veri tabanı, sorgulama ve güncelleme ekranlarından oluşmaktadır. İşlem kaynakları simülatör programı aracılığıyla tanımlanıp simüle edilmektedir. Simülatör ekranları üzerinden girilen bilgiler kullanılarak ISO8583 formatında bir istek mesajı oluşturulur. Simülatör programı işlem sunucusu ile TCP/IP bağlantısı kurarak oluşturduğu istek mesajını işlem sunucusuna gönderir. İşlem sunucusu kendisine gelen her işlem isteği için veri tabanındaki bir programı çalıştırır. Veri tabanı programı işlem üzerinde gereken finansal kontrolleri yaparak işlemin onaylanıp onaylanmayacağına karar verir ve işlem sonucu ile ilgili bilgi döndürür. İşlem sunucusu veri tabanı programının döndürdüğü bilgileri kullanarak ISO8583 formatında bir yanıt mesajı oluşturur ve bu mesajı kurulan bağlantı üzerinden simülatör programına gönderir. Simülatör

programı kendisine gelen yanıt mesajını değerlendirir ve işlem sonucunu ekranda gösterir.

Sorgulama ve güncelleme ekranları, veri tabanındaki kart, müşteri ve işlem bilgilerinin görüntülenmesine ve değiştirilmesine olanak tanımaktadır.

Simülatör programı ile işlem sunucusu arasında iletilen mesajlar ISO8583 standardına uygun formattadır. Veri tabanı bağlantısı için ADO, OLE DB ve ODBC erişim yöntemleri, haberleşme işlemleri için Indy teknolojisi kullanılmıştır.

Çeşitli kaynaklardan yapılan bütün finansal işlemlerde olduğu gibi kredi kartlarında da işlem güvenliği oldukça önemlidir. Kredi kartlarında sağlanan güvenlik mekanizmalarının bir kısmı kuruluşa veya uygulamaya özgüdür, bunların yanı sıra güvenlik ve işlem akışı konusunda uluslararası kuruluşlar tarafından geliştirilmiş birtakım standartlar da vardır. Simülatör programı ve işlem sunucusu arasında gönderilen mesajlar günümüzün en etkin güvenlik mekanizmalarından biri olan SSL protokolü ile şifrelenmektedir.

Sistemin gerçeklenmesinde kullanılan standartlar ve teknolojiler ile ilgili olarak ilerideki bölümlerde daha ayrıntılı bilgi verilecektir.

Bu çalışmada işlem sunucusu ile ilgili performans analizleri de yapılmıştır. Performans analizini yapabilmek için istemci yönetim programı ve istemci program tasarlanmıştır. İstemci yönetim programı istemci programların başlatılmasından sorumludur. İstemci programlar ise işlem örnekleri veri tabanından okudukları kayıtları kullanarak sunucuya istek mesajı göndermektedirler. İşlem sunucusu üzerinde veri tabanı bağlantısı kurma yöntemleri değiştirilerek farklı yük koşulları altında çeşitli ölçümler yapılmıştır. Elde edilen ölçümler kullanılarak grafikler oluşturulmuştur. Grafiklerle ilgili açıklamalar ve her bir yöntemin üstün ve zayıf özellikleri ayrıntılı olarak açıklanmıştır.

2. KAVRAMLAR VE KULLANILAN TEKNOLOJİLER

Bu bölümde uygulama ile ilgili temel kavramlardan ve kullanılan teknolojilerden bahsedilmektedir.

2.1. Kavramlar

Kredi kartları ile para çekme makinaları, satış noktaları, Internet, telefon gibi çeşitli kaynaklardan işlem yapılabilir. İşlem kaynakları bir üye işyerine ait olmak zorundadırlar. Satış noktalarının bağlı oldukları üye işyerleri, satış noktalarının fiziksel olarak bulunduğu ve kullanıldığı yerlerdir. Örnek olarak; mağazalar, marketler gibi. Para çekme makinalarının bağlı oldukları üye işyerleri ise bankalardır.

Üye işyerleri bir finansal kuruma bağlı olarak çalışırlar. Üye işyerinin kullandığı fiziksel cihazlar ve yazılımlar üye işyerinin bağlı olduğu finansal kurum tarafından temin edilir. Bu cihazlar ve yazılımlar üzerinden gönderilen işlem isteklerinin anlık olarak karşılanmasından ve muhasebesinden yine üye işyerinin bağlı olduğu finansal kurum sorumludur.

Çevrimiçi olarak yapılan tüm kredi kartı işlemlerinde, kartı üreten ve finansal olarak işleten kurumdan onay almak zorunludur. Şekil 2.1'de kredi kartı işlem akışı gösterilmektedir.



Şekil 2.1 Kredi kartı işlem akışı

2.2. Kullanılan Teknolojiler

Bu çalışmada veri tabanı fonksiyonları haricindeki tüm programlar (arayüz programı, işlem sunucusu, istemci programlar ve istemci yönetim programı) Delphi programlama dilinde kodlanmıştır. Veri tabanı olarak Oracle 8.1.7 kullanılmıştır. Veri tabanı erişimi, ODBC sürücüsünün OLE DB sağlayıcısı kullanılarak ADO yöntemi ile gerçeklenmiştir. Haberleşme işlemleri için Indy'nin TCP/IP istemci ve sunucu bileşenleri, mesaj formatı için ISO8583 standardı kullanılmıştır. Mesaj güvenliği SSL şifreleme yöntemi ile sağlanmıştır.

3. ISO8583 MESAJ FORMATI

Bu bölümde ISO8583 mesaj formatının amacı ve yapısı anlatılmaktadır.

3.1. Mesaj Formatının Amacı

Finansal işlemler konusunda hizmet veren kuruluşlar, işlemlerini gerçekleştirebilmek için elektronik mesaj alışverişini kullanırlar. Bu mesajların bir kısmı uygulamaya ve kuruluşa özgü olabilmektedir. Uluslararası ISO8583 mesaj standardı, finansal uygulamalar konusunda hizmet veren farklı sistemler ve kuruluşlar arasında mesaj uyumluluğunu sağlamak amacıyla tasarlanmıştır.

3.2. Mesaj Formatının Bileşenleri

ISO8583 standardı, bit haritası kavramını kullanır. Mesajdaki her bir veri elemanı için bit haritasında 1 bitlik yer ayrılmıştır. Bit haritasındaki bitin değeri 1 ise ilgili veri elemanı mesajda mevcut, 0 ise veri elemanı mesajda mevcut değildir. Böylece gerçekleştirilecek olan işlemin tipine göre, sadece gerekli olan veri elemanlarının gönderilmesi sağlanarak, değişken uzunlukta bir mesaj oluşturulmaktadır. Bu özellik, mesaj boyunun kısalması açısından bir üstünlük sağlamaktadır. Şekil 3.1'de mesaj formatının bileşenleri gösterilmektedir.

Mesaj Tipi 4 Byte	Birincil Bit Haritası 8 Byte	İkincil Bit Haritası 8 Byte	Veri elemanları 002 – 064	Veri elemanları 065 – 128

Şekil 3.1 Mesaj formatı

Mesaj Tipi: İşlemin tipi ve nasıl değerlendirileceği hakkında bilgi verir. ISO8583 formatında kullanılan çeşitli mesaj tipleri vardır. Bu mesaj tiplerinden kredi kartı işlemlerinde kullanılanlar ve anlamları Tablo 3.1'de gösterilmektedir.

Tablo 3.1. ISO8583 mesaj tipi ve açıklamaları

ISO8583 Mesaj Tipi	Açıklama
0100	İşlem İsteği
0101	İşlem İsteği Tekrar
0110	İşlem İsteği Yanıt
0120	İşlem Tavsiye İsteği
0121	İşlem Tavsiye İsteği Tekrar
0130	İşlem Tavsiye İsteği Yanıt
0420	İşlem İptal İsteği
0421	İşlem İptal İsteği Tekrar
0430	İşlem İptal İsteği Yanıt
0800	Ağ Yönetim İsteği
0801	Ağ Yönetim İsteği Tekrar
0830	Ağ Yönetim İsteği Yanıt

Bu mesaj tiplerini 3 grupta toplayabiliriz:

İstek Mesajları

Tekrar Mesajları: İsteği gönderen sistem belirlenen bir sürede karşı taraftan yanıt alamazsa tekrar mesajı gönderir. Bu mesajların kodları, istek mesajı+1 olarak belirlenir. İsteği karşılayacak olan sistem, tekrar mesajı geldiğinde istek mesajının işlenip işlenmediğini kontrol eder. İstek mesajı işlenmiş ve yanıt verilmişse; tekrar mesajı dikkate alınmaz yani veri tabanında güncelleme yapılmaz ama karşı tarafa tekrar mesajı için yanıt gönderilir.

Yanıt Mesajları: Bu mesajların kodları, istek mesajı+10 olarak belirlenir. İstek mesajı geldiğinde, isteği karşılayacak olan sistem gerekli kontrolleri yaptıktan sonra karşı tarafa yanıt verir, yanıt kodu bilgisini 39. veri elemanına yazar. Yanıt kodları ISO tarafından belirlenmiş 2 haneli kodlardır. İşlem onaylanmışsa, yanıt veren sistem 6 haneli bir onay kodu üreterek bu bilgiyi 38. veri elemanına yazar.

Birincil Bit Haritası: 8 byte yani 64 bit uzunluğundadır. İlk 64 veri elemanının, mesajda bulunup bulunmadığı hakkında bilgi verir. İlk bit ikincil bit haritasının mevcut olup olmadığını belirtir.

İkincil Bit Haritası: Mesajdaki ilk veri elemanıdır. 8 byte yani 64 bit uzunluğundadır. İkinci 64 veri elemanının (65. veri elemanından 128. veri elemanına kadar), mesajda bulunup bulunmadığı hakkında bilgi verir.

Veri elemanları: Toplam 128 tane veri elemanı vardır. İlk veri elemanı ikincil bit haritasıdır.

4. TCP/IP PROTOKOLLERİ

1990'lı yıllarda TCP/IP, geliştirilen protokollere hem ticari, hem de mimari açıdan baskın gelmiştir. OSI'nin üstünde TCP/IP'nin başarısının birçok nedeni vardır. Bunlardan biri Internet'in TCP/IP protokollerinin üzerine kurulu olmasıdır. Internet'in ve özellikle World Wide Web'in hızlı büyümesi, TCP/IP'nin OSI üzerindeki zaferini pekiştirmiştir.

4.1. TCP/IP Yaklaşımı

TCP/IP protokolleri, haberleşme işleminin tek bir birim tarafından yürütülemeyecek kadar karmaşık olduğu fark edilerek tasarlanmıştır. Buna gore haberleşme işlemleri, diğer sistemlerdeki eş modüllerle haberleşecek alt modüllere ayrılmıştır. Bu sistemdeki bir katman, başka katmanlara hizmet sunarken, diğerlerinin de hizmetlerini kullanır. İyi bir yazılım, bu katmanları, modüler ve hiyerarşik yapıda düzenlemelidir. OSI'de her katmanın kullanılması zorunlu iken, bu kısıtlama TCP/IP'de yoktur. Örneğin, doğrudan IP katmanı ile işlem yapan uygulama katmanı protokolleri mevcuttur.

4.2. TCP ve IP'nin Çalışması

TCP/IP, bir uçbirimin diğer bir uçbirime (farklı bir ağda da olabilir) alt ağ üzerinden veri aktarmasını sağlar. IP, hem uçbirimlerde, hem de yönlendiricilerde yürütülür ve verinin uçbirimler arasında iletimini düzenler. TCP ise sadece uçbirimlerde yürütülür ve veri bloklarının uygun uygulamalar arasında güvenlice teslim edilmesini denetler.

Başarılı bir haberleşme için, tüm sistemde her birimin sistem içinde tek olan global bir adresinin olması gerekir. Bu adres, internet adresi, sayesinde, verinin hedef uçbirime iletilmesi sağlanır. Bunun dışında bir uçbirimdeki her prosesin, uçbirimde tek olan bir numarası olmalı ki, TCP protokolü aldığı veriyi uygun prosese teslim etsin. Bu adrese de port adı verilir.

TCP katmanının temel işlevleri, üst katmandan (uygulama katmanı) gelen bilginin segmentler haline dönüştürülmesi, iletişim ortamında kaybolan bilginin tekrar gönderilmesi ve gelen paketlerin sıralanmasıdır. IP protokolu ise tek tek paketlerin yönlendirilmesinden sorumludur. Bu açıdan bakılırsa küçük ağlarda büyük yükü TCP katmanı yüklenirken, büyük ve karmaşık ağlarda IP katmanı da önemli görevler üstlenir. Göndericinin alıcıya gore hızlı çalıştığı durumlarda, TCP veri iletimini yavaşlatmak için akış kontrol mekanizmaları uygular.

IP, birincil katman 3 Internet protokolüdür. IP, ağ üzerinde yönlendirmenin yanı sıra hata raporlama, veriyi parçalara ayırma ve datagram denilen veri ünitelerini birleştirme gibi işlevler de sağlar. IP adresleri, Ağ Bilgi Merkezi tarafından atanmış 32 bitlik tekil numaralardır. Global olarak tekil olan adresler, IP ağlarının dünyanın neresinde olursa olsun birbirleri ile haberleşebilmelerini sağlar.

IP adresi 3 parçaya ayrılmıştır. Birinci parça ağ adresini, ikinci parça alt ağ adresini ve üçüncü parça da makina adresini temsil eder. IP adresleme 3 farklı ağ sınıfını destekler. A sınıfı ağlar az sayıda çok büyük ağlar için kullanılmaktadır çünkü ağ adres alanı için sadece 8 bit sağlarlar. Ağ adres alanı için, B sınıfı ağlar 16 bit ve C sınıfı ağlar ise 24 bit yer ayırırlar. C sınıfı ağlar makina alanı için sadece 8 bit sağlarlar bu yüzden ağ başına düşen makina sayısı sınırlıdır. Her 3 sınıfta da en soldaki bit ağ sınıfını gösterir.

5. INDY

Bu bölümde Indy teknolojisi ve özellikleri hakkında bilgi verilmektedir.

5.1. Indy Teknolojisi

Internet Direct (Indy), popüler Internet protokollerinden oluşan, açık kodlu bir bileşen paketidir. Indy bileşenleri Delphi 6 ve Kylix programlama ortamlarının içinden paket halinde çıkmaktadır.

Indy, tamamen engelleyici soket'ler uzerine kurulmus olup Borland firmasının ürünleri (Delphi, C++ Builder ve Kylix) için tasarlanmış bir bileşen paketidir. İstemci bileşenlerin kullanımı kolaydır, çünkü programcı işlemlerini belirli bir sıra dahilinde yazar ve sunucu bileşenler çok görevcikli halde çalışırlar.

5.2. Indy Paketinde İstemci, Sunucu Mimarisinin Gerçeklenmesi

İstemci özelliğinin gerçekleştirilmesi için TIdTCPClient bileşeni kullanılır.

Bir TIdTCPClient gerçekleştirimi:

TIDTCPClient = class(TIdTCPConnection)

TIdTCPClient bir TCP istemcisinin tüm özelliklerini taşır. Birçok diğer Indy istemci bileşenleri TIdTCPClient sınıfından türetilmiş sınıflardır. Örnek: TIdDayTime, TIdEcho, TIdFinger, TIdFTP, TIdGopher, TIdFTTP, TIdNNTP, TIdPOP3, TIdQUOTD, TIdSMTP, TIdTelnet ve TIdWhois.

Sunucuya bağlantı açılması:

TIdTCPClient.Connect

Connect prosedürü istemci ile sunucu arasında bir bağlantı kurmaya çalışan prosedürdür. Bu prosedür bağlantı kurabilmek için Host özelliğinde geçerli bir sunucu ve Port özelliğinde de bir iskele numarası olmasını gerektirir.

Sunucu özelliğinin gerçekleştirilmesi için TIdTCPServer bileşeni kullanılır.

Çok görevcikli bir TCP sunucusunun gerçekleştirimi:

TIdTCPServer = class(TIdComponent)

TIdTCPServer, çok görevcikli bir TCP sunucusunun tüm özelliklerini taşır. TIdTCPServer istemci bağlantılarını dinlemek için ayrı bir görevcik yaratır. Ve bir Görevcik Yöneticisi ile beraber çalışarak her yeni gelen istemci bağlantısı için ayrı bir görevcik yaratır. TIdTCPServer diğer bir çok TCP sunucusu için ana sınıf (base class) olarak kullanılabilir.Bir çok diğer Indy sunucu bileşenleri TIdTCPServer sınıfından türetilmiştir. Örnek: TIdChargenServer, TIdDayTimeServer, TIdDICTServer, TIdEchoServer, TIdFingerServer, TIdGopherServer, TIdHostNameServer, TIdHTTPServer, TIdIRCServer, TIdNNTPServer, TIdQUOTDServer, TIdTelnetServer ve TIdWhoisServer.

5.3. Engelleyici Soket Yapısı

Indy engelleyici soketleri kullanır. Engelleyici soketleri kullanmak bir dosya okuma ya da yazma işlemi yapmak gibidir. Veriyi okuduğunuzda ya da yazdığınızda çağırmış olduğunuz fonksiyon işlem bitene kadar dönmez. Dosyalarla çalışmaktan farkı, verinin yapılmak istenen okuma ya da yazma işlemi için kısa zaman içinde hazır olmayabileceğidir. Engelleyici soket işlemleri en fazla bilgisayar ağı ya da modem'in hızı kadar hızlı olabilir.

Örneğin, bağlantı kurma işlemi basitçe Connect metodunu çağırır ve o metodun dönmesini bekler. Bağlantı kurma işlemi başarıyla sonuçlandıysa metod döner, aksi halde hata yaratır.

5.3.1. Engelleyici soketler

Engelleyici soketlerin diğer asenkron çalışan soketlere göre kullanışsız olduğu yönünde genel bir inanış söz konusudur. Fakat bu inanış yanlıştır. "Winsock" Windows platformuna taşındığı zaman ilk aşamada bir problem ortaya çıkmıştı. Unix işletim sisteminde genellikle yeni süreç yaratmak için "fork" fonksiyonu kullanılır. Bu fonksiyon görevcik yaratmaz fakat onun yerine ayrı bir süreç yaratır. Unix istemcileri fork fonksiyonu ile engelleyici soket kullanan süreçler yaratabilirler.

Windows 3.x fork fonksiyonunu kullanamıyordu dolayısı ile çok görevcikli bir işletim sistemi değildi. Engelleyici soketler, kullanıcı arayüzlerinin kilitlenmesine ve programın yanıt vermemesine yol açıyordu. Bu problemi gidermek için Winsock'a asenkron eklentiler yapıldı. Bu şekilde artık programın ana ve tek görevciği kilitlenmiyordu. Fakat bu eklentiler farklı bir programlama yöntemi gerektiriyordu. Bu sebeple piyasada engelleyici soketlerin kullanışsız ve kötü olduğu yolunda fikirler yayılmaya başladı.

Daha sonra gerçek çok görevcikli bir işletim sistemi olan Win32 ortaya çıktı. Fakat bu anda piyasadaki fikirler çoktan yerleşmişti. Böylece engelleyici soketler hakkındaki fikirler devam etti.

Gerçekte, engelleyici soketler Unix'te kullanılan tek soket çeşididir. Ayrıca engelleyici soketlerin başka bir çok avantajı daha vardır. Çok görevcikli programlamaya daha uygundur, güvenlikleri daha kolay sağlanabilir. Sonradan Unix işletim sistemine de asenkron soketler için eklentiler yapıldı fakat bu eklentiler hiçbir zaman genel kullanım kazanamadı. Şu an Unix işletim sisteminde engelleyici soketler nerdeyse her alanda kullanılıyor ve kullanılmaya devam edecek gibi görünüyor.

5.3.2. Engelleyici soketlerin avantajları

- Programlama kolaylığı – Engelleyici soketleri programlamak çok kolaydır. Programların bütün kodları aynı yerde bulunur ve sıralı olacak şekilde yazılır.

- Unix İşletim Sistemi'ne taşınma kolaylığı – Unix engelleyici soketleri kullandığı için taşınabilir program yazmak daha kolaydır. Bu durumun farkında olarak Indy bileşenleri hem Windows hem de Unix işletim sistemleri için aynı kaynak koduna sahiptir.

- Görevciklerle beraber çalışmaya uygunluk – Engelleyici soket kullanan programlar sıralı bir şekilde yazıldığı için görevciklerle beraber çok rahat kullanılabilirler.

5.3.3. Engelleyici soketlerin dezavantajları

- İstemci programlardaki kullanıcı arayüzünün işlem süresince kilitlenmesi – Engelleyici soketler yapılmak istenen işlem tamamlanmadan dönmezler. Böyle bir işlem programın ana görevciğinde yapılmak istenirse, uygulama kullanıcı arayüzü mesajlarını işleyemez. Bu sebeple kullanıcı arayüzü işlem bitene kadar kilitlenmiş, donmuş olarak görülür, çünkü program ekran güncelleme, ekranı tekrar boyama gibi kullanıcı arayüzü işlemlerini yapamamaktadır.

5.4. TIdAntiFreeze Bileşeni

Indy, kullanıcı arayüzü kilitlenme problemini çözen bir bileşene de sahiptir. Engelleyici soket kullanan uygulamaya TIdAntiFreeze bileşeninden bir tane konursa, kullanıcı arayüzü işlemleri herhangi bir kilitlenme olmadan işlenebilir.

TIdAntiFreeze bileşeni belirli aralıklarla engelleyici soket çağrılarını kendi içinde zaman aşımına uğratır ve bu zaman aşımı süresi boyunca Application.ProcessMessages fonksiyonunu çağırır. Bu fonksiyon programın ana mesaj döngüsüdür ve kullanıcı arayüzü ile ilgili yapılan bütün işlemler bu döngü içerisinde işlenir.

5.5. Çok Görevcikli Yapı

Çok görevcikli programlar engelleyici soketler ile birlikte neredeyse her zaman kullanılır. Engelleyici olmayan soketler için de çok görevcikli programlar kullanılabilir fakat bu durumda yapılması gereken fazladan işler çok görevcikliliğin avantajlarının ortadan kaybolmasına neden olur.

5.5.1. Çok görevcikliliğin avantajları

- Öncelik verme – Her görevciğin önceliği ayrı ayrı değiştirilebilir. Bu şekilde belirli bazı sunucu işlerine ya da belirli bazı bağlantılara daha fazla ya da az CPU zamanı ayrılabilir.

- Kapsama – Her bağlantı kendi içindedir ve diğer bağlantılarla karışma ya da kesişme olasılığının en az olması sağlanmıştır.

- Güvenlik – Her görevciğin güvenlik ayarları ayrı ayrı yapılabilir.

- Çoklu işlemciler – Çok görevciklilik otomatik olarak çoklu işlemci avantajının kullanılmasını sağlar.

- Serileşme yoktur – Çok görevciklilik gerçek anlamda paralel işleme özelliğini getirir. Çok görevciklilik olmasaydı, her istek tek bir görevcik tarafından işlenmek zorunda kalacaktı. Bu yöntemin işe yaraması için her işlemin hızlı şekilde çalışan küçük bölümlere ayrılması gerekir. Eğer işlemin herhangi bir bölümü diğerlerini engelleyici bir özelliğe sahipse ya da bu bölümün tamamlanması uzun sürüyorsa diğer bütün işlemler onun bitmesini beklerler. Bu şekilde de bütün işlemlerin sıralı bir şekilde işlenmesi gerekir. Oysa çok görevciklilik sayesinde her işlem ayrı bir bütün olarak tanımlanır ve sistem CPU zamanını bu işlemler arasında dağıtır.

5.5.2. Görevcik havuzlama

Görevciklerin yaratılması ve yokedilmesi çok kaynak harcayan işlemlerdir. Bu özellikle çok kısa süren istemci bağlantılarına sahip olan sunucular için geçerlidir. Bu tür sunucular bir görevcik yaratırlar, ve bu görevcik çok kısa süreyle çalışıp işini bitirdikten sonra da onu yokederler. Böylece çok sık şekilde görevcik yaratılıp yokedilme durumuyla karşılaşılır. Bu duruma örnek olarak bir zaman sunucusu ya da web sunucusu gösterilebilir. Bu tür sunuculara istemciler tarafından sadece tek bir istek gönderilir ve bu isteğin karşılığı olarak tek bir yanıt beklenir. Bir tarayıcı kullanarak bir Internet sitesinde gezinirken, bu sunucuya aynı anda yüzlerce bağlantı isteği geliyor olabilir.

Görevcik havuzlama, bu gibi durumları önleyebilir. Görevcikleri istek üzerine yaratıp yok etmektense, daha önceden yaratılmış bir görevcik havuzundaki aktif olmayan görevciklerden birisi ödünç alınabilir. Alınan görevcik yapacağı işlemi bitirdiyse yok edilmek yerine görevcik havuzuna geri verilir. Görevcikler havuzdayken “Aktif değil” olarak işaretlenir, dolayısı ile fazladan CPU gücü harcamazlar. Daha da iyisi bu havuzun boyutu sistemin ihtiyaçlarına uygun olarak dinamik olarak değiştirilebilir.

Indy görevcik havuzlama özelliğini destekler. Bu özellik Indy'nin TIdThreadMgrPool bileşeni kullanılarak yapılır.

Meşgul bir sunucuda yüzlerce hatta binlerce görevciğe aynı anda ihtiyaç duyulabilir. Yüzlerce sayıda görevciğin sistemi çok yoracağı şeklinde yanlış bir inanış vardır. Bir çok sunucuda görevcikler ömürlerini çoğunlukla veri için bekleyerek geçirirler. Bu şekildeki engelleyici fonksiyon çağrılarını bekleyen görevcikler aktif durumda değildirler. Dolayısı ile 500 görevciği olan bir sunucunun aslında aynı anda sadece 50 görevciği aktif durumda olabilir.

5.5.3. Görevcikler ve global kısım

Birden fazla görevcik aynı anda aynı veriye okuma ya da yazma için erişmeye çalışıyorsa, erişimin bir şekilde kontrollü bir halde yapılması gerekir. Verinin bütünlüğünü korumak için bu tür bir kontrollü erişim kesinlikle gereklidir. Çok görevcikli program yazmaya alışkın olmayan programcılarda bu durum bir zorluğa sebep olabilir. Fakat, sunucuların bir çoğu global veriye ihtiyaç duymaz. Çoğu sunucu ayrıştırılmış fonksiyonlarla çalışır. Yani her görevcik izole edilmiş ayrı bir görevi gerçekleştirir.

5.6. Indy Metodolojisi

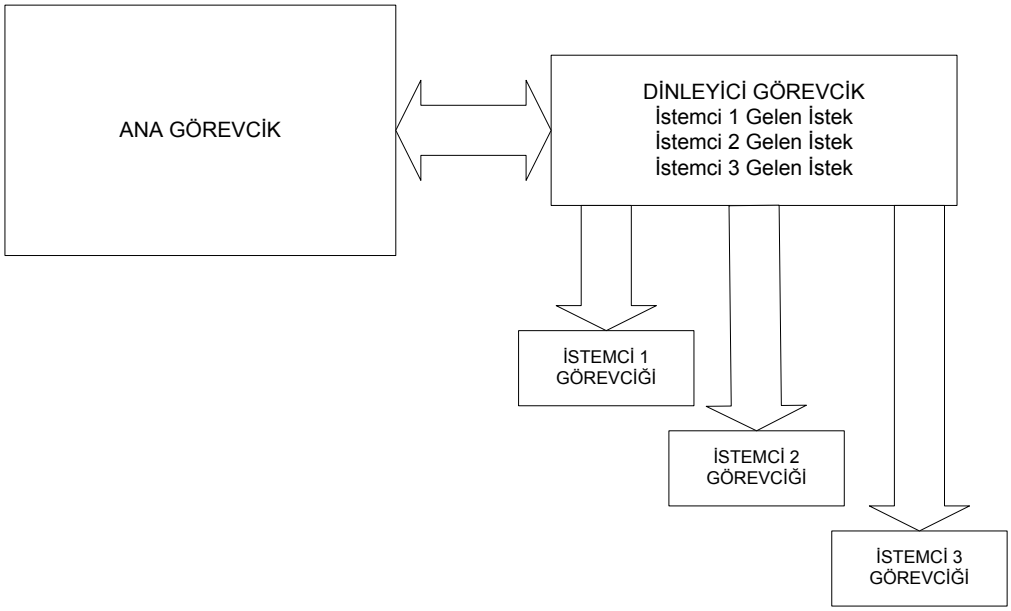
Indy, iyi bilinen Winsock bileşenlerinden faklı bir yapıdadır. Hemen hemen bütün diğer bileşenler asenkron soketleri kullanıyorken, Indy engelleyici soketleri kullanır. Diğer bileşenler programcının çalışma anında oluşan olaylar için kod yazmasına, durum makinaları kurmalarına ve belirli bir şeyi bekleyen döngüler gerçekleştirmesine sebep olurlar. Fakat Indy'de bir sunucuya bağlanılmak istendiğinde basitçe Connect metodu çağırılır ve dönmesi beklenir. Indy ile çalışmak dosyalarla çalışmak gibidir, ayrıca Indy'nin yapısı çok görevcikli uygulamalar için de çok uygundur.

Indy baştan sona kadar çok görevcikli olacak şekilde tasarlanmıştır. Indy istemcileri ve sunucuları yazmak Unix istemci ve sunucuları yazmaya çok benzerdir, fakat onların aksine yazması çok daha kolaydır. Çünkü Indy ve Delphi doğrudan yığın çağrılarını çağırmanızı önleyen bir soyutlama tabakası sağlar. Unix'te ise programcı doğrudan yığın çağrılarını kullanır.

Unix sunucularının tipik olarak bir yada birden fazla dinleyici süreçleri vardır. Bu süreçler istemcilerden gelecek olan bağlantıları bekler. Servis bekleyen her istemci için fork fonksiyonu kullanılarak yeni bir süreç yaratılır. Bu şekilde programlamak çok kolaydır çünkü yaratılan her süreç tek bir istemciden sorumludur. Ayrıca süreç kendi güvenlik bağlamı içinde çalışır.

Indy sunucuları da benzer şekilde çalışırlar. Windows'ta Unix'in aksine fork kullanımı iyi değildir fakat görevcik kullanımı iyi çalışır. Indy sunucuları her istemci için ayrı bir görevcik yaratırlar.

Indy sunucuları programın ana görevciğinden farklı bir dinleyici görevcik yaratırlar. Dinleyici görevcik gelecek olan istemci bağlantılarını dinler. Her istemci isteği için ayrı bir görevcik yaratır. Şekil 5.1'de Indy sunucusunun çalışma yöntemi gösterilmektedir.



Şekil 5.1 Indy sunucusunun çalışma yöntemi

Tipik bir Indy istemcisi ise aşağıdaki gibidir:

```
With IndyClient do begin
    Connect;
    Try
```

```
        // Burada yapılması gereken işlem yapılır.
      Finally
        Disconnect;
      End;
End;
```

6. ADO VERİ ERİŞİM YÖNTEMİ

Bu bölümde ADO, OLE DB ve ODBC veri erişim yöntemleri ile ilgili bilgi verilmektedir.

6.1. ODBC, OLE DB ve ADO

ODBC ve OLE DB, veriye ulaşmak için kullanılan Microsoft Windows'un uygulama programlama arayüzleridir. Daha eski ODBC spesifikasyonu, birincil ilişkisel SQL veri tabanlarına erişimi sağlar. Microsoft'un bir sonraki kuşak veri erişim spesifikasyonu olan OLE DB, daha geniş veri sağlayıcı kümesine erişime olanak tanır. Bu sağlayıcılar, klasik ilişkisel veri tabanı sistemlerinin yanı sıra ilişkisel olmayan veri tabanı sistemlerini ve elektronik posta sistemlerini de kapsar. OLE DB, ODBC'nin yerini almaz. Doğrusu, OLE DB, ODBC veri kaynakları ile birlikte kullanılabilen bir veri sağlayıcıyı içerir.

ODBC, tek bir uygulamanın aynı kaynak kod ile farklı veri tabanı yönetim sistemlerine erişmesini sağlar. Veri tabanı uygulamaları, sürücü denilen veri tabanına özgü modülleri kullanan ODBC arayüzündeki fonksiyonları çağırırlar. Sürücülerin kullanımı, yazıcı sürücülerinin kelime işleyen programları yazıcıya özgü komutlardan arındırması gibi, uygulamaları veri tabanına özgü çağrılardan arındırılar. Sürücüler çalışma anında yüklendikleri için, kullanıcının yeni bir veri tabanına bağlanması için yeni bir sürücü eklemesi yeterlidir, uygulamayı yeniden derlemesine gerek yoktur.

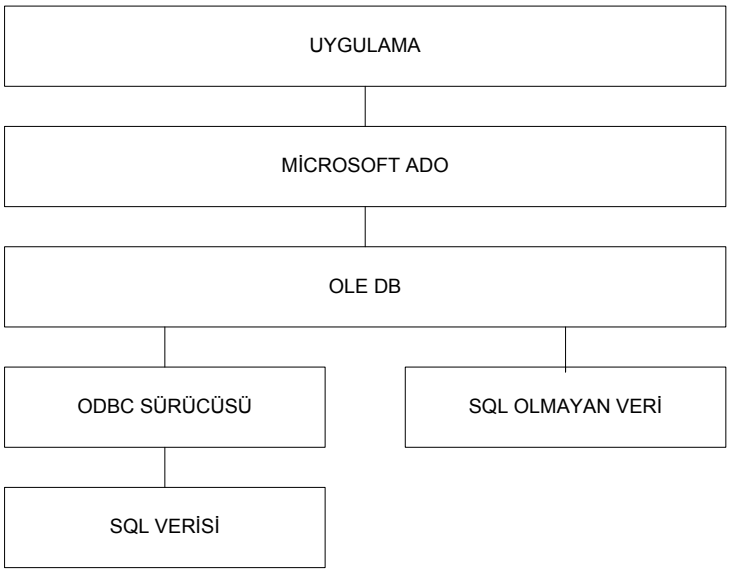
OLE DB, veri erişimi için arayüz kümesidir ve Microsoft'un veri tipinden bağımsız olarak evrensel veri entegrasyonu sağladığı bileşen veri tabanı mimarisidir. OLE DB, esnek ve verimli bir veri tabanı mimarisi sağlar. OLE DB, kayıtları saklamak ve getirmek için kullanılan temel COM yapı bloğudur. OLE DB, tüm veri tiplerine erişmek ve işletmek için arayüzler tanımlar. Bu arayüzler veriyi kullanan uygulamaların yanı sıra veri tabanı sağlayıcıları tarafından da kullanılırlar.

ADO (ActiveX Veri Nesneleri), OLE DB için yüksek düzeyli bir arayüzdür. ADO, OLE DB'nin ODBC veri sağlayıcısı aracılığıyla, ODBC veri kaynaklarına erişmek için kullanılabilir. ADO programlama modeli, Microsoft'un mevcut veri erişim modelleri içinde en iyisidir. Daha önceki modellerde olan kullanışsız ve hantal işler ADO yönteminde düzeltilmiş veya ortadan kaldırılmıştır. ADO nesneleri, tüm veri kaynağı türlerine erişim için hızlı, kolay ve üretken bir arayüz sağlarlar. ADO, 1'den n kata kadar istemci/sunucu uygulamaları ve veri tabanına erişen Web tabanlı yazılımlar için basit bir veri arayüzüdür. ADO modeli, bir yandan alt seviyede kullandığı veri sağlayıcısının tüm yapabildiklerini sağlarken, diğer yandan da genel operasyonlar için kısayollar sunar. Gelecekte ADO, elektronik posta sistemlerine veya diğer ilişkisel olmayan veri sağlayıcılarına erişmek için de kullanılabilecektir.

Microsoft ADO, aslında OLE DB'nin üzerine kurulmuştur. OLE DB güçlüdür ama programlaması zordur. ADO, OLE DB'nin karmaşıklığını gidermiştir. ADO, Microsoft teknolojisi olduğu için en iyi performansı Microsoft Access ve Microsoft SQL Server ile gösterir. Özet olarak ADO, Windows üzerinde çalışan ve veri tabanı olarak Microsoft'un ürünlerini kullanan veya diğer veri tabanı sunucularının OLE DB sağlayıcılarını kullanan uygulamalarda üstünlük sağlar.

6.2. ADO Mimarisi

Şekil 6.1, ADO'nun OLE DB ve ODBC ile olan kullanılabilirliğini göstermektedir.

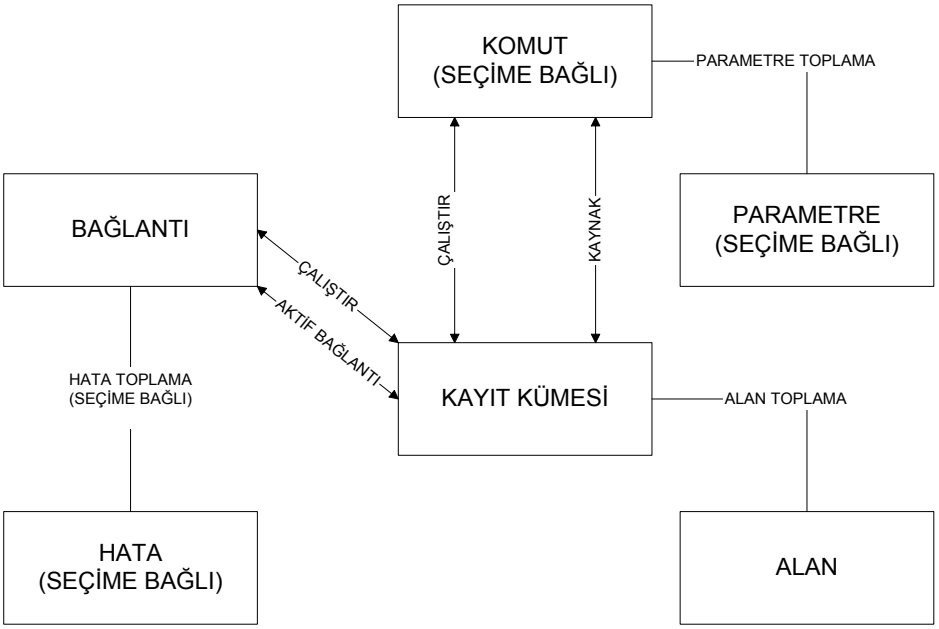


Şekil 6.1 ADO mimarisi

Şekil 6.1'den de görüldüğü üzere, ADO, OLE DB veri sağlayıcısı ile doğrudan veya ODBC ile dolaylı olarak veriye ulaşabilir. ADO'nun mevcut veri tabanı mimarisine bir katman daha eklediği tartışılabilir ama performansı iyidir.

6.3. ADO Programlama Modeli

Şekil 6.2, ADO modelindeki nesneleri ve birbirleri ile olan ilişkilerini göstermektedir.



Şekil 6.2 ADO programlama modeli

- Bağlantı Nesnesi

Bağlantı nesnesi veri kaynağına olan bağlantıyı temsil eder ve komutların çalıştırılmasına izin verir. Herhangi bir komutu çalıştırmak için, bağlantı nesnesinin "Çalıştır" metodu kullanılır. Eğer komut kayıtlar döndürüyorsa, bir "Kayıt Kümesi" nesnesi yaratılır ve döndürülür.

- Komut Nesnesi

Veri kaynağı tarafından işlenebilecek bir komutu (sorgu veya ifade olarak da bilinir) temsil eder. Komutlar, basit SQL ifadeleri veya veri tabanındaki saklı prosedür çağrıları olabilirler. Komutlar, komut nesnesinin "Çalıştır" metodu ile çalıştırılırlar. Veri sağlayıcı komut parametrelerini destekliyorsa, komutlar

parametreler ile birlikte çalıştırılabilirler. Her parametre için bir parametre nesnesi yaratılır.

- Parametre Nesnesi

Komutun parametresini temsil eder.

- Kayıt Kümesi Nesnesi

Veri tabanından kayıt getirmek için kullanılır. Bu nesne imleç fonksiyonlarını gerçekleştirir.

- Alan Nesnesi

Kayıt kümesindeki veri alanlarını temsil eder.

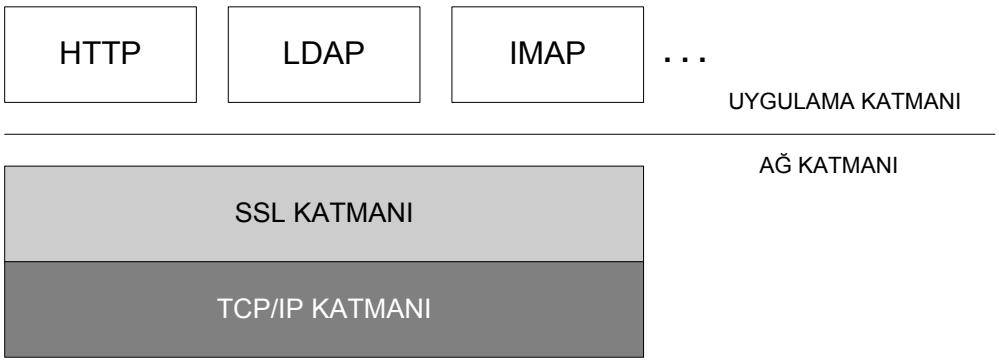
- Hata Nesnesi

Veri kaynağından döndürülen hatayı temsil eder.

7. SSL GÜVENLİK MEKANİZMALARI

TCP/IP protokolü, Internet üzerinde verinin iletilmesini ve yönlendirilmesini sağlayan protokoldür. HTTP, LDAP yada IMAP gibi diğer protokoller TCP/IP'nin üzerine kurulmuşlardır. Bu protokoller, Web sayfalarını göstermek ya da elektronik posta sunucusu işlemlerini yürütmek için TCP/IP'yi kullanırlar.

Şekil 7.1'de görüldüğü üzere, SSL, TCP/IP'nin üzerinde, diğer üst seviye uygulama protokollerinin altında yer alır.



Şekil 7.1 SSL'in TCP/IP ve uygulama protokolleri ile ilişkisi

SSL, daha üst seviyeli protokoller adına TCP/IP'yi kullanır. Bu işi yaparken, SSL destekli bir sunucunun kendisini SSL destekli bir istemciye ve istemcinin kendisini sunucuya doğrulamasını sağlar ve her iki makina arasında şifrelenmiş bir bağlantı gerçekleştirir. Bu yetenekler Internet ve diğer TCP/IP ağları üzerindeki veri iletişiminin temel gereksinimlerini sağlar.

- SSL sunucu doğrulama: Bir kullanıcının, sunucunun kimliğini onaylamasını sağlar. SSL destekli istemci yazılımı, standart açık anahtar şifreleme tekniklerini kullanarak, bir sunucunun sertifika ve genel kimliğinin geçerli olduğunu ve istemcinin güvenilir CA (Sertifika Otoritesi) listesindeki bir CA tarafından verilmiş olduğunu kontrol edebilir. Örneğin, kullanıcının kredi kartı bilgilerini bir bilgisayar ağı üzerinden gönderdiği ve verileri alan

sunucunun kimliğini kontrol etmek istediği bir durumda bu tür bir doğrulama çok önemlidir.

- SSL istemci doğrulama: Bir sunucunun, kullanıcının kimliğini onaylamasını sağlar. Sunucu doğrulamada kullanılan aynı tekniklerle, SSL destekli sunucu yazılımı, bir istemcinin sertifika ve genel kimliğinin geçerli olduğunu ve sunucunun güvenilir CA listesindeki bir CA tarafından verilmiş olduğunu kontrol edebilir. Örneğin, bir bankanın gizli finansal bilgilerini bir müşterisine gönderdiği ve alıcının kimliğini kontrol etmek istediği bir durumda bu tür bir doğrulama çok önemlidir.

- Şifrelenmiş SSL bağlantısı: Yüksek derecede gizlilik sağlamak için, istemci ve sunucu arasında gönderilen tüm bilginin gönderen yazılım tarafından şifrelenmesi ve alan yazılım tarafından da şifresinin çözülmesini gerektirir. Gizlilik, özel işlemler söz konusu olduğunda her iki taraf için de çok önemlidir. Ayrıca şifrelenmiş SSL bağlantısı üzerinden gönderilen veri, iletim sırasında değiştirilme işlemine karşı bir mekanizma ile de korunur.

SSL protokolü 2 alt protokol içerir: SSL kayıt protokolü ve SSL el sıkışma protokolü. SSL kayıt protokolü, veri iletim formatını tanımlayan protokoldür. SSL el sıkışma protokolü ise SSL destekli bir istemci ile SSL destekli bir sunucunun SSL kayıt protokolü ile tanımlanmış formatta veri alışverişi yapabilmeleri için ilk bağlantı kurdukları zaman kullanılır. İlk bağlantı aşamasında karşılıklı gönderilip alınan mesajlardan oluşur. Bu mesaj alışverişi aşağıdaki özellikleri sağlayacak şekilde tanımlanmıştır:

- Sunucuyu istemciye doğrulama

- İstemci ve sunucunun kullanacakları şifreleme algoritmasını seçebilmelerine olanak sağlama

- Seçimli olarak istemciyi sunucuya doğrulama

- Açık anahtarlı şifreleme tekniklerini kullanarak paylaşılan gizli bilgiler üretme

- Şifreli bir SSL bağlantısı kurma

7.1. SSL ile Kullanılan Şifrelemeler

SSL protokolü, bir çok farklı şifreleme algoritması kullanımını destekler. SSL protokolü, şifreleme algoritmalarını, istemci ve sunucuyu birbirlerine karşı doğrularken, sertifikaları iletirken ve oturum anahtarlarını oluştururken kullanır. Kullanılan SSL versiyonu, kabul edilebilir şifreleme gücüyle ilgili şirket politikaları, SSL destekli yazılımın ihracıyla ilgili hükümet kısıtlamaları gibi faktörlere bağlı olarak istemci ve sunucular değişik şifreleme algoritmaları kullanabilirler. Diğer fonksiyonlarının yanında SSL el sıkışma protokolünün bir görevi de, istemci ve sunucunun birbirlerini doğrularken, sertifikaları iletirken ve oturum anahtarlarını oluştururken hangi şifreleme algoritmalarını kullanacaklarını belirlemektir.

- DES: Veri Şifreleme Standardı, ABD hükümeti tarafından kullanılan bir şifreleme algoritması.
- DSA: Dijital İmza Algoritması, ABD hükümeti tarafından kullanılan sayısal doğrulama standardının bir parçası olan algoritma.
- KEA: Anahtar Değişim Algoritması, ABD hükümeti tarafından anahtar değişimi için kullanılan algoritma.
- MD5: Rivest tarafından geliştirilen "Mesaj Özeti" algoritması.
- RC2 ve RC4: RSA veri güvenliği için geliştirilen Rivest şifrelemesi.
- RSA: Rivest, Shamir ve Adleman tarafından hem şifreleme hem de doğrulama amacıyla geliştirilmiş bir açık anahtar algoritması.
- RSA Anahtar Değişimi: SSL tabanlı bir anahtar değiştirme algoritması.
- SHA-1: Güvenli Hash Algoritması, ABD hükümeti tarafından kullanılan bir hash fonksiyonu.
- Triple_DES: 3 kez uygulanan DES.

KEA ve RSA gibi anahtar değişim algoritmaları, istemci ve sunucunun SSL oturumu boyunca kullandıkları simetrik anahtarları belirlerler. En çok kullanılan SSL paketleri RSA değişim algoritmalarını kullanırlar.

SSL 2.0 ve SSL 3.0 protokolleri üst üste binen birden fazla şifreleme paketi kullanımını destekler. Sistem yöneticileri, istemci ve sunucular için bu şifreleme paketlerini istedikleri gibi aktif hale getirebilir yada kullanımdan çıkarabilirler. Bir istemci ve sunucu, SSL el sıkışma sırasında bilgi alışverişinde bulunduğunda sahip oldukları en güçlü şifreleme paketi üzerinde anlaşırlar ve SSL oturumu boyunca bunu kullanırlar. Bir organizasyonun şifreleme paketi seçimi, verinin hassasiyeti, şifreleme paketinin hızı ve ihraç kurallarının uygulanabilirliği gibi çeşitli faktörlere bağlıdır.

Bazı organizasyonlar, daha zayıf şifreleme yapan SSL bağlantılarını engellemek için daha zayıf şifreleme paketlerini kullanımdan kaldırmak isteyebilirler. Fakat ABD hükümeti, 40-bitten daha fazla şifreleme yapan yazılımlar üzerinde kısıtlamalar getirmiştir ve 40-bit şifrelemeyi destekleyenlerin sadece ABD'de bulunan ağ tarayıcılarına erişimi kısıtlanmıştır. Bu sebeple mümkün olduğunca fazla kullanıcıya hizmet verebilmek için, sistem yöneticileri çok sayıda şifreleme paketinin aktif olarak kullanımda kalmasını sağlamalıdırlar. Bu durumda yurtiçinden gelen bir istemciye yanıt veren sunucu en güçlü şifre paketini kullanırken, bir yurtdışı istemcisi ile ilgilenen sunucu ABD hükümetinin izin verdiği sınırlar içerisindeki bir şifre paketini kullanabilecektir. Fakat 40-bitlik şifrelemeler göreceli olarak kırılabileceği için mümkün olan durumlarda sistem yöneticileri 40-bitlik şifre paketlerini kullanımdan kaldırmalıdırlar.

7.2. SSL El Sıkışma

SSL protokolü, açık anahtar ve simetrik anahtar şifrelemenin bir kombinasyonunu kullanır. Simetrik anahtar şifrelemesi açık anahtarlı şifrelemeye göre çok daha hızlıdır, fakat açık anahtarlı şifreleme daha iyi doğrulama teknikleri sağlar. Bir SSL oturumu her zaman "SSL el sıkışma" adı verilen bir mesaj alışverişi ile başlar. Bu mesaj alışverişi sunucunun açık anahtar tekniklerini kullanarak kendisini istemciye tanıtmasını/doğrulamasını sağlar. Daha sonra istemci ve sunucu açılacak olan oturumda hızlı şifreleme ve şifre çözme için kullanılacakları simetrik anahtarları yaratırlar. Seçimli olarak, bu mesaj alışverişi istemcinin kendisini sunucuya doğrulamasını da sağlar.

SSL el sıkışma sırasındaki mesaj alışverişi aşağıdaki şekilde özetlenebilir:

1. İstemci sunucuya, kendi SSL versiyon numarasını, şifreleme ayarlarını, rasgele üretilmiş veri ve sunucunun kendisiyle haberleşmesinde ihtiyaç duyacağı diğer bilgileri gönderir.

2. Sunucu istemciye, kendi SSL versiyon numarasını, şifreleme ayarlarını, ragele üretilmiş veri ve istemcinin kendisiyle haberleşmesinde ihtiyaç duyacağı diğer bilgileri gönderir. Ayrıca sunucu kendi sertifikasını gönderir ve istemci, sunucudan doğrulama gerektiren herhangi bir kaynak isteğinde bulunursa istemcinin sertifikasını ister.

3. İstemci sunucudan gelen bilgilerin bir kısmını kullanarak sunucuyu doğrular. Eğer sunucu doğrulanamazsa, kullanıcı uyarılır ve şifrelenmiş ve doğrulanmış bir bağlantının kurulamadığı söylenir. Sunucu başarılı bir şekilde doğrulanırsa istemci 4. adımdan devam eder.

4. İstemci, el sıkışmada şu ana kadar üretilen veriyi kullanarak (kullanılan şifreleme paketine bağlı olarak sunucunun yardımıyla) oturum için asıl gönderilecek paketten önceki "gizli ön bilgiyi" üretir. Bu bilgiyi, sunucunun açık anahtarı ile şifreler (sunucunun açık anahtarı 2. adımda gönderilen sertifikadan elde edilir) ve şifrelenmiş bilgiyi sunucuya gönderir.

5. Sunucu eğer istemci doğrulaması isteğinde bulunduysa (el sıkışmada seçime bağlı bir adımdır) istemci sadece bu el sıkışmaya özel, hem istemci hem de sunucu tarafından bilinen bir veri parçası daha yaratır ve imzalar. Daha sonra istemci imzalanmış veriyi ve kendi sertifikasını şifrelenmiş "gizli ön bilgi" ile birlikte sunucuya gönderir.

6. Eğer sunucu istemci doğrulaması isteğinde bulunmuşsa, sunucu istemciyi doğrulamaya çalışır. İstemci doğrulanamazsa oturum sonlandırılır. Eğer istemci başarılı bir şekilde doğrulanmışsa sunucu kendi kapalı anahtarını kullanarak "gizli ön bilginin" şifresini çözer, daha sonra belirli bazı adımları izleyerek (aynı gizli ön bilgiyi kullanarak istemci de aynı adımları izler) "asıl bilgiyi" üretir.

7. Hem istemci, hem de sunucu "asıl bilgiyi" kullanarak oturum anahtarlarını üretirler. Oturum anahtarları, SSL oturumu sırasında iletilen veriyi şifrelemek ve

şifresini çözmek için ve ayrıca veri üzerinde iletim sırasında değişiklik olup olmadığını anlamak için kullanılırlar.

8. İstemci sunucuya bir mesaj göndererek, sunucuyu bundan sonraki mesajların oturum anahtarları ile şifreleneceği konusundan haberdar eder. Daha sonra el sıkışmanın istemci kısmının sona erdiğini belirten ayrı bir şifrelenmiş mesaj gönderir.

9. Sunucu da istemciye bundan sonraki mesajların oturum anahtarları ile şifreleneceğini bildiren bir mesaj atar. Daha sonra da el sıkışmanın sunucu tarafının da sona erdiğini belirten ayrı bir şifrelenmiş mesaj atar.

10. Artık SSL el sıkışması bitmiş ve SSL oturumu başlamıştır. İstemci ve sunucu bu andan itibaren birbirlerine gönderdikleri veriyi şifrelemek, şifresini çözmek ve verinin değişik kontrolünü yapmak için oturum anahtarlarını kullanırlar.

İstemci ve sunucu doğrulama işlemi sırasında belirli bir parça verinin açık-kapalı anahtar ikilisinin birisi ile şifrelenirken, diğeri ile şifresinin çözüldüğüne dikkat edilmelidir.

Sunucu doğrulaması durumunda, istemci gizli ön bilgiyi sunucunun açık anahtarı ile şifreler. Sadece ilgili kapalı anahtar bilginin şifresini çözebilir. Bu şekilde istemci sadece bağlı olduğu sunucunun şifrelenmiş veriyi çözebileceğinden emin olmuş olur. Aksi taktirde sunucu gizli ön bilginin şifresini çözemez ve simetrik anahtarları yaratamazdı. Bu durumda da oturum kapatılırdı.

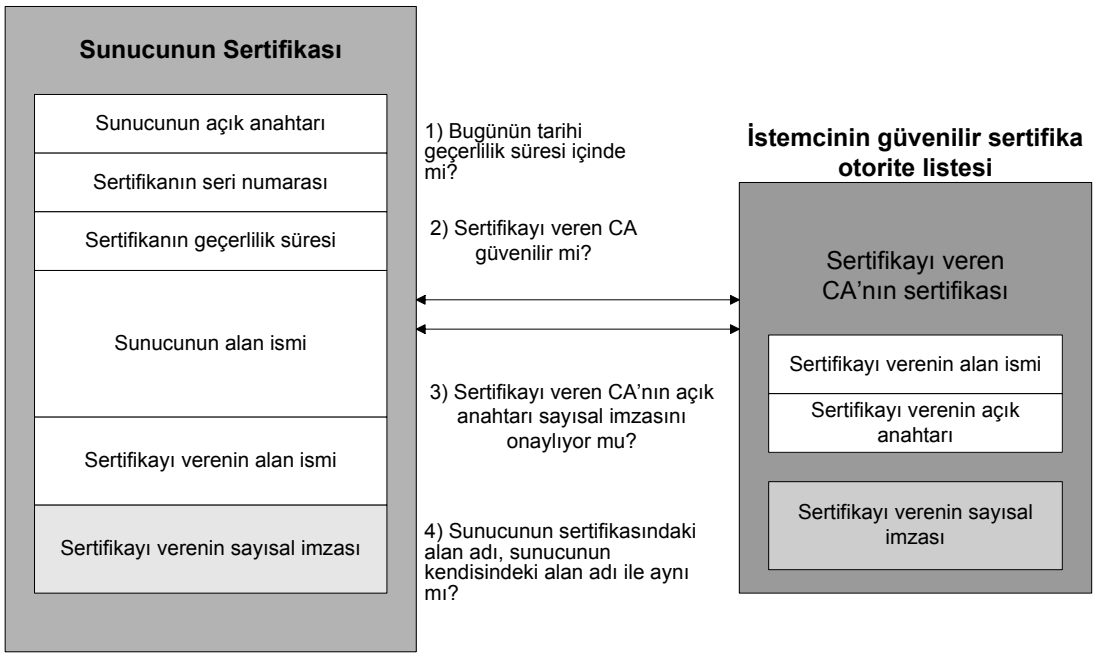
İstemci doğrulaması durumunda, istemci rasgele bir veriyi kendi kapalı anahtarı ile şifreler, yani sayısal bir imza üretir. Bu imzanın doğrulamasını sadece istemcinin sertifikasındaki açık anahtarı yapabilir. Aksi taktirde sunucu sayısal imzayı doğrulayamazdı ve oturum kapatılırdı.

7.3. Sunucu Doğrulama

SSL el sıkışmasının 2. adımında da açıklandığı gibi, sunucu kendisini doğrulamak için istemciye bir sertifika gönderir. İstemci 3. adımda bu sertifikayı kullanarak

sertifikanın temsil ettiğini iddia ettiği sunucuyu gerçekten temsil edip etmediğini anlar.

Açık anahtar ile bu açık anahtarı içeren sertifika tarafından tanımlanan sunucu arasındaki bağlantıyı doğrulamak için, SSL kullanan istemci Şekil 7.2'deki dört soruya "Evet" yanıtını vermelidir. Her ne kadar 4. soru teknik olarak SSL'in bir parçası değilse de bu işlem istemcinin sorumlulukları arasındadır.



Şekil 7.2 İstemcinin sunucu sertifikasını doğrulaması

SSL kullanan istemci, sunucunun kimliğini doğrulamak için aşağıdaki adımları izler:

1. Bugünün tarihi geçerlilik süresi içinde mi? İstemci, sunucu sertifikasının geçerlilik süresini kontrol eder. Eğer bugünün tarih ve zamanı bu tarih aralığınının içinde değilse, doğrulama işlemi sona erer. Eğer bugünün tarih ve zamanı geçerlilik süresi içinde ise istemci doğrulama işlemine 2. adımdan devam eder.

2. Sertifikayı veren CA güvenilir mi? SSL kullanan her istemci, güvenilir CA'lerin bir listesini tutar. Bu liste istemcinin hangi sunucu sertifikalarını kabul edeceğini belirler. Eğer sertifikayı veren CA'in ismi bu listedeki CA'lerin herhangi birisiyle uyuşursa, bu sorunun yanıtı "Evet" tir ve istemci 3. adımdan devam eder. Eğer CA listede yoksa sunucu doğrulanamamış olur.

3. Sertifikayı veren CA'in açık anahtarı sayısal imzasını onaylıyor mu? İstemci, CA'in sertifikasında bulunan sayısal imzayı onaylamak için yine CA'in sertifikasında bulunan açık anahtarı kullanır. Eğer sunucu sertifikasındaki bilgi CA tarafından imzalandıktan sonraki sürede değişmişse ya da CA'in sertifikasındaki açık anahtar CA'in sunucu sertifikasını imzalamakta kullandığı gizli anahtar ile uyuşmuyorsa, istemci sunucuyu doğrulayamaz. Eğer CA'in sayısal imzası onaylanabiliyorsa, sunucu, kullanıcının sertifikasını CA'den gelen geçerli bir "tanıtım mektubu" olarak kabul eder ve işleme devam eder. Bu noktada istemci, sunucunun sertifikasının geçerli olduğu sonucuna varmıştır. 4. ve 5. adımları izlemek istemcinin sorumluluğundadır.

4. Sunucunun sertifikasındaki alan adı, sunucunun kendisindeki alan adı ile aynı mı? Bu adım sunucunun gerçekten sunucu sertifikasındaki alan adıyla belirtilen ağ adresi ile aynı yerde yerleştirilmiş olduğunu onaylar. Bu adım teknik olarak SSL protokolünün bir parçası olmamasına rağmen, istemciler bu adımı izlemeli ve alan adları uymazsa sunucuyu doğrulamayı ve bağlantı kurmayı reddetmelidirler. Eğer sunucunun gerçek alan adı ile sunucu sertifikasındaki alan adları uyuşursa istemci 5. adımdan devam eder.

5. Sunucu doğrulanır. İstemci SSL el sıkışması ile devam eder. Eğer istemci herhangi bir sebeple 5. adıma kadar gelememişse sertifika tarafından tanımlanan sunucu doğrulanamaz, kullanıcı problemden haberdar edilir ve şifreli ve doğrulanmış bir bağlantı kurulamadığı belirtilir. Eğer sunucu, istemci doğrulama gerektiriyorsa, İstemci doğrulama bölümünde anlatılan adımları izlemelidir.

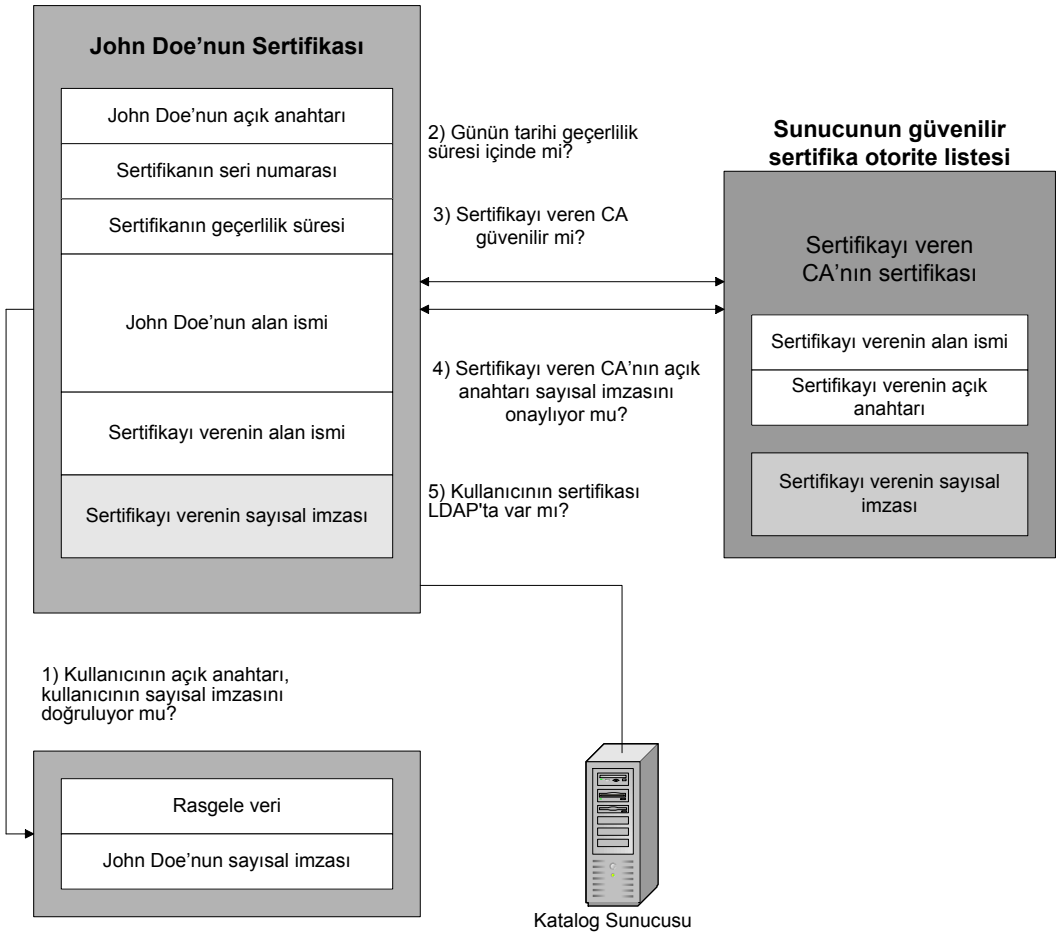
Yukarıdaki adımları geçen sunucu, istemcinin SSL el sıkışması'nın 4. adımında göndermiş olduğu Gizli Önbilgi'nin şifresini kendi gizli anahtarını kullanarak başarılı bir şekilde çözebilmelidir.

7.4. İstemci Doğrulama

SSL kullanan sunucu, istemci doğrulamayı gerektirecek şekilde konfigüre edilebilir. Sunucu bu şekilde tasarlanmışsa, istemci sunucuya hem sertifikayı hem de kendini doğrulaması için sayısal olarak imzalanmış veri parçasını gönderir. Sunucu, sayısal

olarak imzalanmış veriyi, sertifikadaki açık anahtarı onaylamak ve sertifikanın temsil ettiği kimliği doğrulamak için kullanır.

Açık anahtar ve açık anahtarı tutan sertifika tarafından kimliği tespit edilen istemci arasındaki bağlantıyı doğrulamak için, SSL kullanan sunucu Şekil 7.3'te gösterilen ilk 4 soru için "evet" yanıtını almalıdır. 5. soru SSL protokolünün bir parçası olmamasına rağmen, Netscape sunucuları kullanıcının bir LDAP dizinine de girebilmesi için bu gereksinimi destekleyecek şekilde konfigüre edilebilirler. Şekil 7.3 bir Netscape sunucusunun istemci sertifikasını nasıl doğruladığını göstermektedir.



Şekil 7.3 Sunucunun istemci sertifikasını doğrulaması

SSL kullanan bir sunucu, bir kullanıcının kimliğini doğrulamak için aşağıdaki adımları izler:

1. Kullanıcının açık anahtarı, kullanıcının sayısal imzasını doğruluyor mu? Sunucu, sertifikadaki açık anahtar ile kullanıcının sayısal imzasının doğrulanabildiğini

kontrol eder. Eğer doğrulanabiliyorsa, sunucu John Doe'ya ait olduğu iddia edilen açık anahtar ile imzayı oluşturmada kullanılan kapalı anahtarın birbirine uyduğunu ve verinin imzalandıktan sonra bozulmadığını kanıtlar.

Bu noktada, anahtar ve sertifikada belirtilen DN arasındaki bağlantı henüz kurulmamıştır. Sertifika, kullanıcıyı taklit etmeye çalışan biri tarafından da yaratılmış olabilir. Açık anahtar ve DN arasındaki bağlantıyı onaylamak için sunucu 3. ve 4. adımları da tamamlamalıdır.

2. Günün tarihi geçerlilik süresi içinde mi? Sunucu sertifikanın geçerlilik süresini kontrol eder. Günün tarihi geçerlilik süresinin dışında ise, doğrulama süreci daha fazla ilerlemez. Geçerlilik süresinin içinde ise sunucu 3.adıma geçer.

3. Sertifikayı veren CA güvenilir mi? Her SSL kullanan sunucu, Şekil 7.3'ün sağ tarafında taranmış bölge ile temsil edilen güvenilir sertifika listesini tutar. Bu liste sunucunun hangi sertifikaları kabul edeceğini belirler. Sertifikayı üretenin DN'i suncunun üzerindeki güvenilir CA listesindeki bir CA'in DN'ine uyarsa, bu sorunun yanıtı evettir ve sunucu 4. adıma geçer. Sertifikayı üreten CA listede değilse, istemci doğrulanmayacaktır.

4. Sertifikayı veren CA'in açık anahtarı sayısal imzasını onaylıyor mu? Sunucu, sertifikada bulunan CA'in sayısal imzasını onaylamak için yine CA'in sertifikasında bulunan açık anahtarı kullanır. Eğer sertifikadaki bilgi CA tarafından imzalandıktan sonra değişmişse yada CA sertifikasındaki açık anahtar CA'in sertifikayı imzalamak için kullandığı gizli anahtar ile uyuşmuyorsa, sunucu kullanıcının kimliğini doğrulamayacaktır. Eğer CA'in sayısal imzası onaylanabiliyorsa sunucu kullanıcının sertifikasını CA'den gelen geçerli bir "tanıtım mektubu" olarak kabul eder ve devam eder. Bu noktada SSL protokolüne göre sunucu istemciyi doğrulanmış kabul eder ve 6. adımda anlatılan bağlantı ile devam eder.

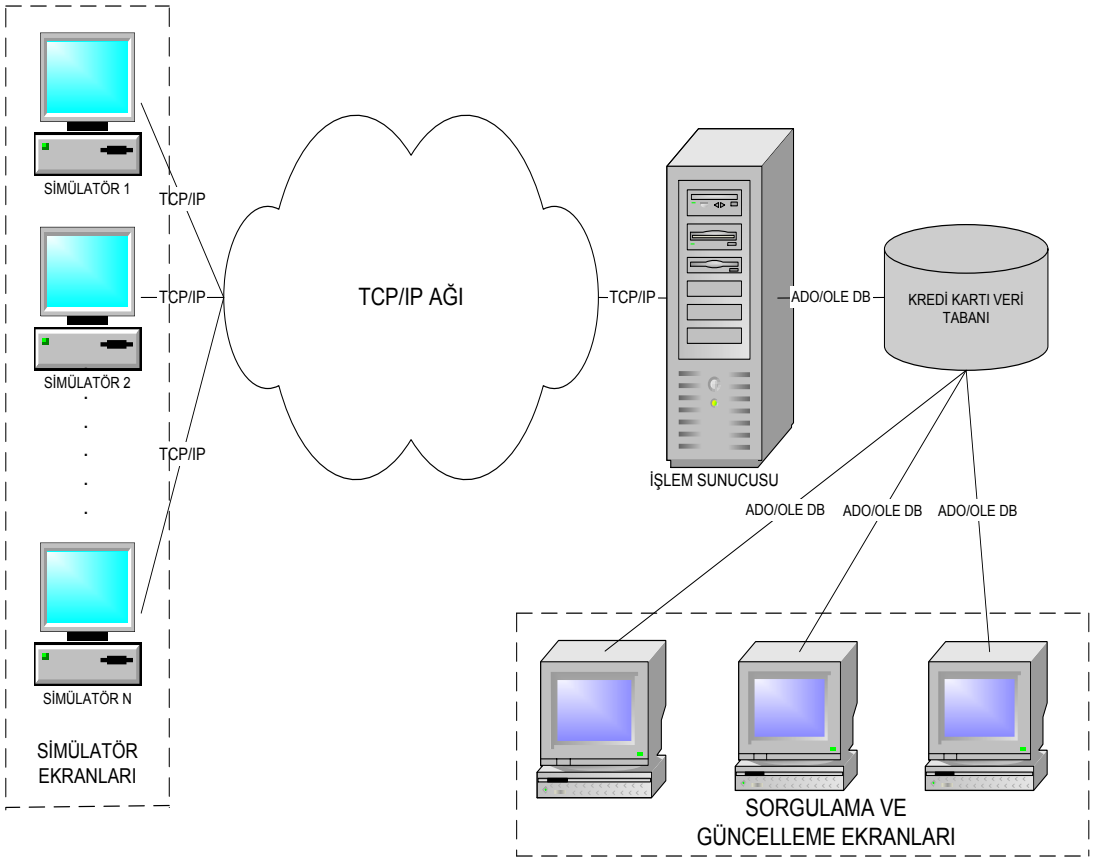
5. Kullanıcının sertifikası LDAP'ta var mı? Bu seçimlik adım, sistem yöneticisine kullanıcının sertifikasını, diğer adımlardaki tüm testleri geçse bile, iptal etmek için bir yöntem sunar. Netscape Sertifika Sunucusu iptal edilmiş bir sertifikayı otomatik olarak LDAP'tan siler. Bundan sonra, bu adımı gerçekleştirmek üzere ayarlanmış tüm sunucular, o sertifikayı doğrulamayı ve bir bağlantı kurmayı

reddedeceklerdir. Eğer LDAP'taki kullanıcı sertifikası ile SSL El Sıkışması'ndaki kullanıcı sertifikası aynı ise sunucu 6. adımdan devam eder.

6. Doğrulanmış sunucunun istek yaptığı kaynaklara erişim yetkisi var mı? Sunucu, Erişim Yetki Listeleri'ne bakarak istemcinin hangi kaynaklara erişim yetkisi olduğunu kontrol eder ve uygun erişimi içeren bir bağlantı kurar. Eğer sunucu herhangi bir sebeple 6. adıma gelememişse, sertifikanın tanımladığı kullanıcı doğrulanamamış olur ve kullanıcının doğrulama gerektiren sunucu kaynaklarına erişimi engellenir.

8. SİSTEM BİLEŞENLERİ VE ÇALIŞMA MANTIĞI

Şekil 8.1'de görüldüğü üzere sistem 4 modülden oluşmaktadır. Modüllerin teknik tasarımı ve çalışma mantığı bu bölümde detaylı olarak açıklanmıştır.



Şekil 8.1 Sistem bileşenleri

Simülatör ekranları ile sorgulama ve güncelleme ekranları Şekil 8.2'deki Kredi Kartı İşlem Sistemi Arayüz Programı üzerinden kullanılmaktadır.



Şekil 8.2 Kredi Kartı İşlem Sistemi Arayüz Programı

8.1. Simülatör Ekranları

İşlem kaynaklarının simüle edilebilmesi sağlanır. Simülatör ekranları Şekil 8.3'te gösterildiği üzere kaynak tanımlama ekranları ve işlem simülasyon ekranlarından oluşmaktadır. Kaynak tanımlama ekranları kullanılarak finansal kurum, üye işyeri ve cihaz tanımları yapılabilmektedir. İşlem simülasyon ekranları ile kaynak simülasyonu ve mesaj simülasyonu olmak üzere iki tür simülasyon gerçekleştirilir. Simülatör programı Delphi programlama dilinde gerçeklenmiştir. Veri tabanı bağlantıları için ADO veri erişim yöntemi, haberleşme işlemleri için Indy'nin istemci bileşenleri kullanılmıştır. İşlem simülasyon ekranlarından gönderilen mesajların güvenliği SSL şifreleme mekanizması ile sağlanmıştır.



Şekil 8.3 Simülatör ekranları menüsü

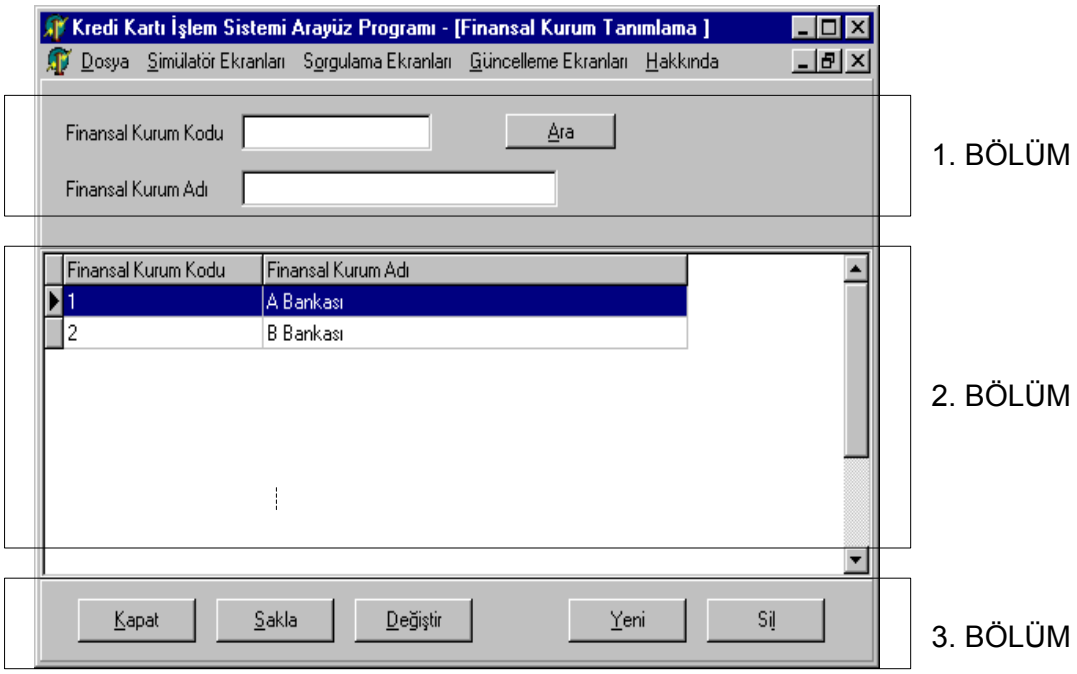
8.1.1. Kaynak tanımlama ekranları

İşlem kaynaklarını tanımlamak amacıyla kullanılır. Kaynak tanımlama ekranları menüsü Şekil 8.4'te gösterilmektedir.



Şekil 8.4 Kaynak tanımlama ekranları menüsü

Tanım ekranları aşağıdaki standartlara uygun olarak tasarlanmıştır. Ekranlar Şekil 8.5'te gösterildiği üzere 3 bölümden oluşmaktadır:



Şekil 8.5 Tanımlama ekranlarının bölümleri

1.bölüm : Ekranın en üst kısmında yer alan bölümdür. Bilgi girişini sağlayan kutular ve Ara düğmesinden oluşmaktadır.

Ara : Veri tabanındaki istenilen kayıda ulaşmak için kullanılır.

2.bölüm : Ekranın orta kısmında yer almaktadır. Bilgilerin listelendiği bölümdür.

3.bölüm : Ekranın en alt kısmında yer almaktadır. Bilgi girişi ve güncellemesini sağlayan düğmelerden oluşmaktadır. Bu düğmeler ve işlevleri sırasıyla şöyledir:

Kapat : Ekranın kapatılmasını sağlar.

Sakla : Girilen veya değiştirilen bilginin veri tabanına kayıt edilmesini sağlar.

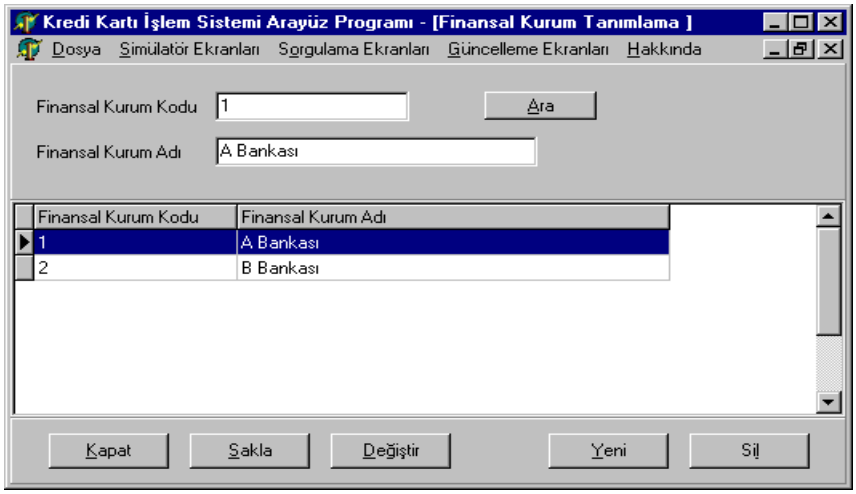
Değiştir : Veri tabanındaki bilginin değiştirilmesini sağlar.

Yeni : Veri tabanına bilgi girişini sağlar.

Sil : Veri tabanındaki bilginin silinmesini sağlar.

8.1.1.1. Finansal kurum tanım ekranı

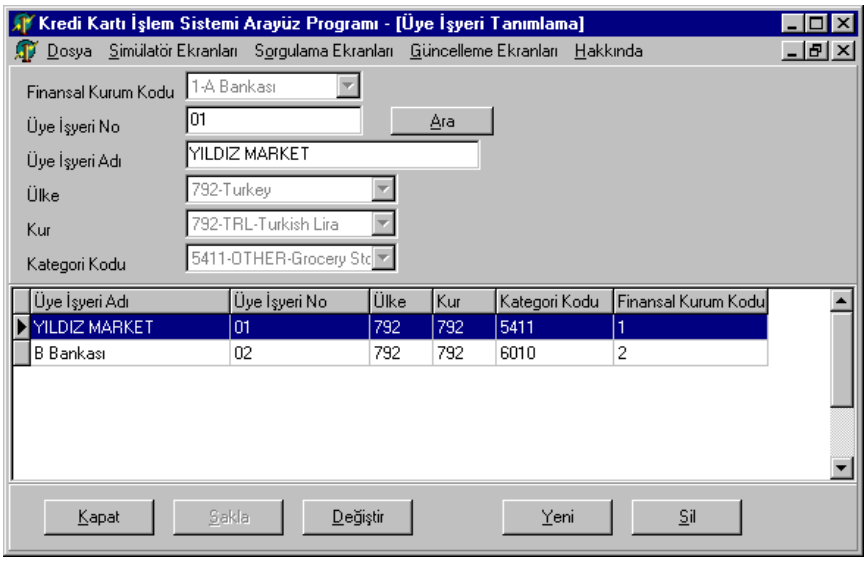
İşlemin yapıldığı üye işyerinin bağlı olduğu finansal kurumun tanımlanması için kullanılır. Şekil 8.6'da Finansal Kurum Tanım Ekranı gösterilmektedir.



Şekil 8.6 Finansal Kurum Tanım Ekranı

8.1.1.2. Üye işyeri tanım ekranı

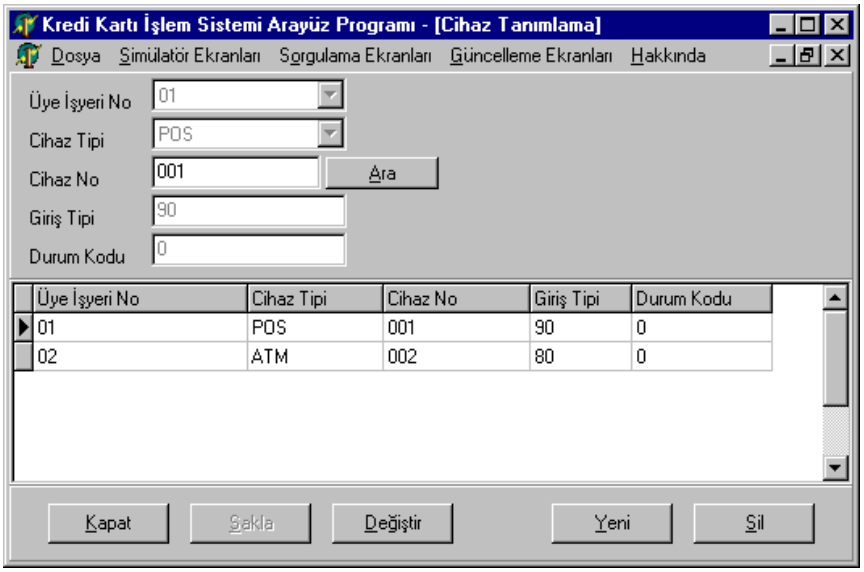
İşlemin yapıldığı üye işyerinin tanımlanması için kullanılır. Üye işyeri bir finansal kuruma bağlı olmak zorundadır dolayısıyla tanımlama yapılırken üye işyerinin bağlı olduğu finansal kurum seçilir. Ayrıca üye işyerinin bulunduğu ülke kodu, kullandığı para birimi ve kategori kodu bilgileri girilir. Şekil 8.7'de Üye İşyeri Tanım Ekranı gösterilmektedir.



Şekil 8.7 Üye İşyeri Tanım Ekranı

8.1.1.3. Cihaz tanım ekranı

İşlemin yapıldığı fiziksel cihazın tanımlanması için kullanılır. Cihazın bağlı olduğu üye işyeri seçilir. Cihazın tipi ATM yada POS olmak zorundadır. Cihazın giriş tipi ve durum kodu bilgileri girilir. Şekil 8.8'te Cihaz Tanım Ekranı gösterilmektedir.



Şekil 8.8 Cihaz Tanım Ekranı

8.1.2. İşlem simülasyon ekranları

Kredi kartı işlem isteği yaratmak için gereken bilgiler simülasyon ekranlarından girilir. Bu ekranların çalışma mantığı Şekil 8.9'da gösterilen algoritma ile özetlenebilir:



Şekil 8.9 İşlem simülasyon ekranlarının çalışma mantığı

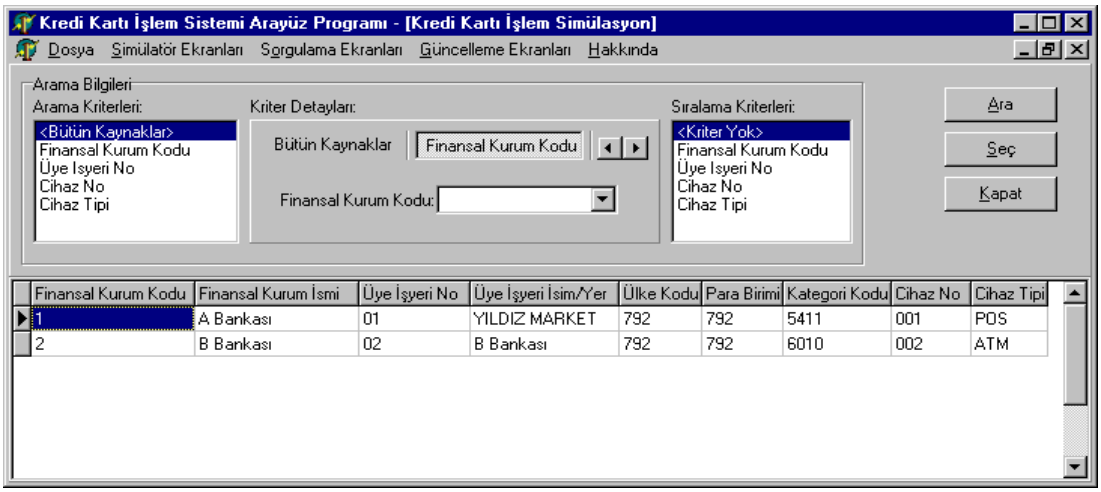
Simülasyon ekranları kaynak simülasyonu ve mesaj simülasyonu olmak üzere iki ekrandan oluşmaktadır. Şekil 8.10'da İşlem Simülasyon Ekranları Menüsü gösterilmektedir.



Şekil 8.10 İşlem Simülasyon Ekranları Menüsü

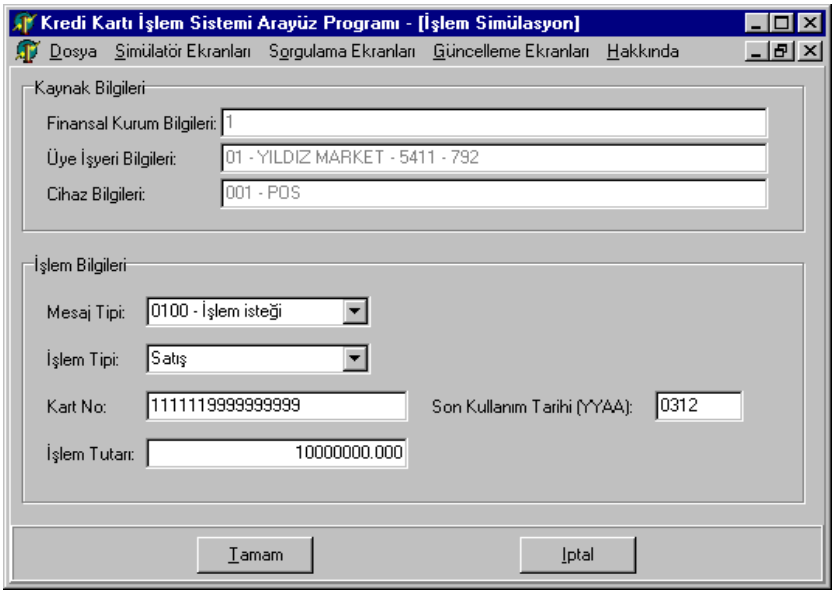
8.1.2.1. Kaynak simülasyonu

İşlem Simülasyon Ekranları menüsünden Kaynak Simülasyonu seçeneği seçilir ve Kaynak Liste Ekranı açılır. Şekil 8.11'de görülen Kaynak Liste Ekranından çeşitli kriterlere göre sorgu yapılır ve neticesinde istenen kritelere uygun olan kaynaklar listelenir.



Şekil 8.11 Kaynak Liste Ekranı

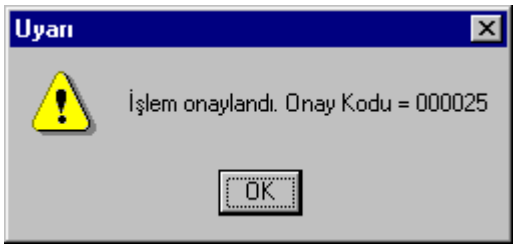
Bu kaynaklardan 1 tanesi seçilir. Kaynak ile ilgili bilgiler Şekil 8.12'de görülen İşlem Simülasyon ekranına aktarılır. Bu ekrandan işlem ile ilgili diğer bilgiler girilir ve Tamam düğmesine basılınca bir işlem isteği oluşturulup, işlem sunucusuna yönlendirilir.



Şekil 8.12 İşlem Simülasyon Ekranı

İşlem sunucusundan gönderilen yanıt bir uyarı mesajı olarak ekranda gösterilir.

Örnek olarak işlem onaylanmışsa, onay kodu ile birlikte Şekil 8.13'deki gibi bir uyarı mesajı verilir.



Şekil 8.13 İşlemin onaylanması durumunda verilen uyarı mesajı

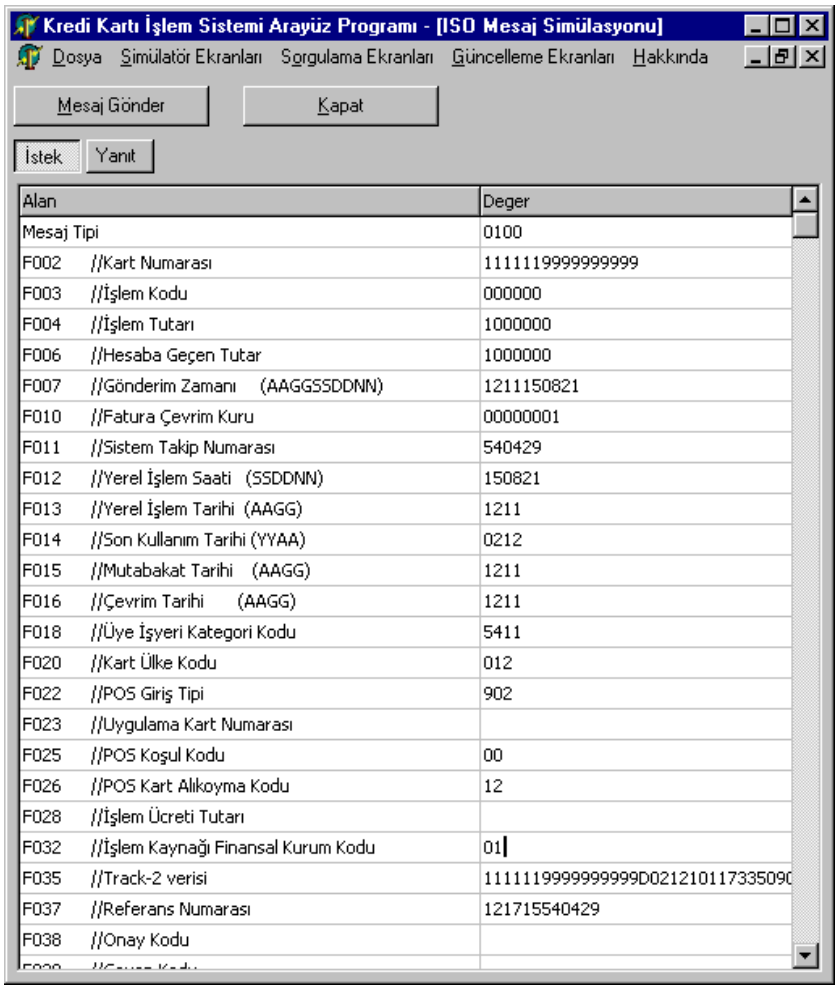
İşlem onaylanmamışsa da Şekil 8.14'te görüldüğü gibi onaylanmama sebebi ile ilgili bilgi verilir.



Şekil 8.14 İşlemin onaylanamaması durumunda verilen uyarı mesajı

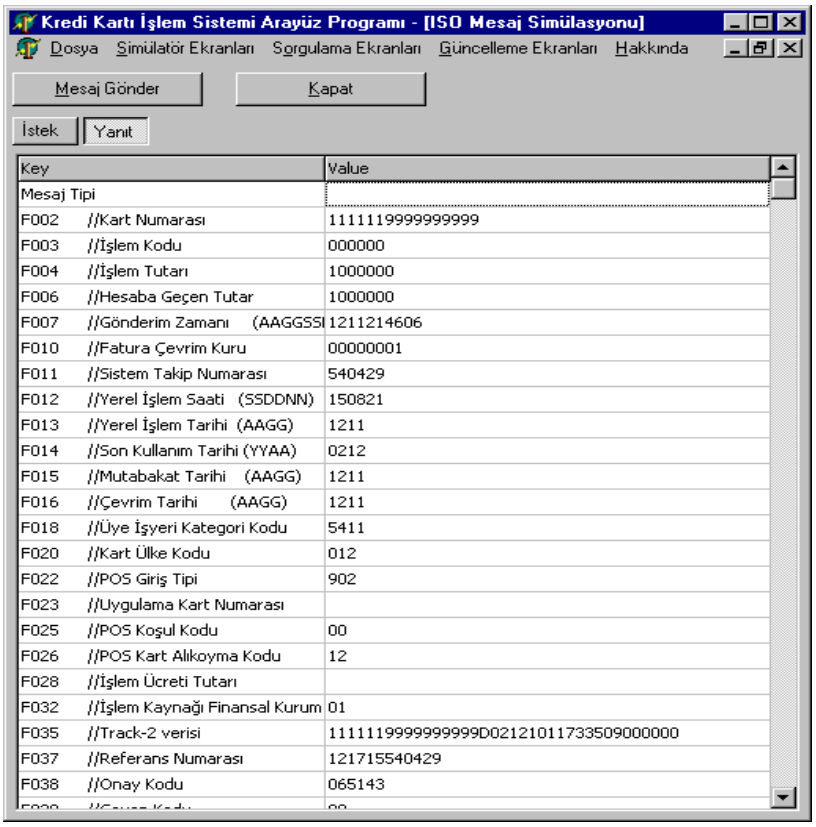
8.1.2.2. Mesaj simülasyonu

Simülatör programından mesaj simülasyonu seçeneği seçilir. ISO8583 mesaj formatındaki mesaj alanlarının listelendiği, Şekil 8.15'de görülen Mesaj Simülasyonu ekranı açılır. Ekran üzerinde İstek seçeneği seçili iken, işlem için gerekli olan alanlar doldurulur.



Şekil 8.15 Mesaj Simülasyon Ekranı – İstek Mesajı

Ekrandan Tamam düğmesine basılınca bir mesaj isteği oluşturulur ve bu istek işlem sunucusuna yönlendirilir. İşlem sunucusunun gönderdiği yanıt mesajı, Şekil 8.16'de görüldüğü üzere Mesaj Simülasyon ekranında Yanıt seçeneği seçili iken gösterilir.



Şekil 8.16 Mesaj Simülasyon Ekranı – Yanıt Mesajı

8.2. İşlem Sunucusu

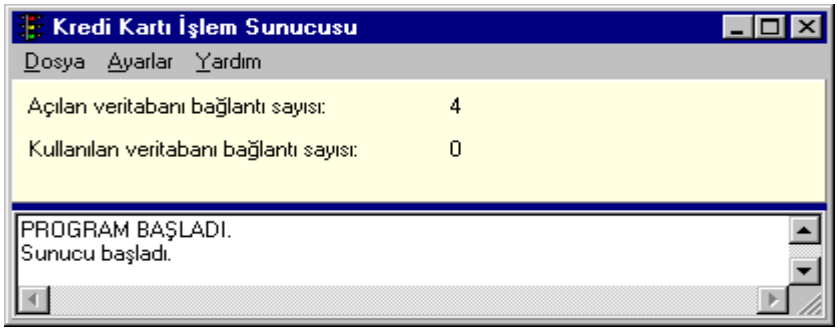
8.2.1. İşlem sunucusunun başlatılması

İşlem sunucusunun başlama mantığı Şekil 8.17'deki algoritma ile özetlenebilir.



Şekil 8.17 İşlem sunucusu başlatılma mantığı

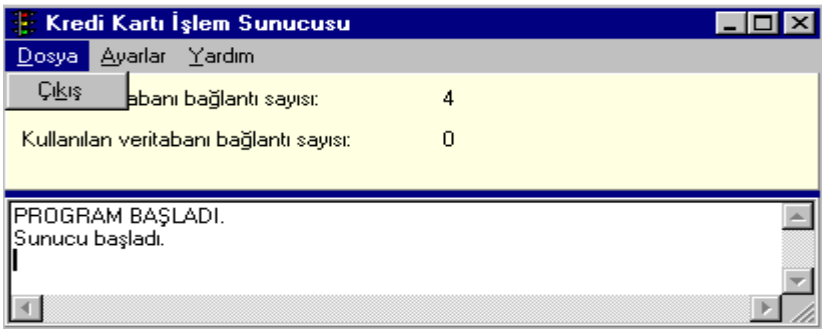
Şekil 8.18'de işlem sunucusunun başlangıç durumu gösterilmektedir.



Şekil 8.18 İşlem sunucusunun başlangıç durumu

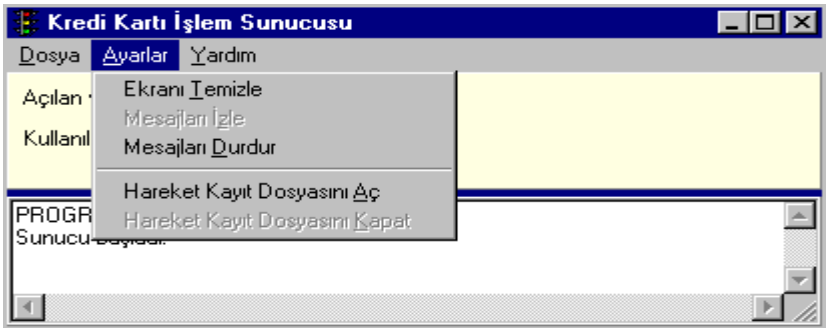
8.2.2. İşlem sunucusunun menüleri

Şekil 8.19'da Dosya menüsü gösterilmektedir. Dosya menüsünde Çıkış seçeneği vardır. Çıkış seçildiğinde dinleyici görevcik inaktif yapılır, yani sunucu durdurulur. Sunucu ve kayıt nesneleri yok edilir.



Şekil 8.19 İşlem Sunucusu Dosya Menüsü

Şekil 8.20'de Ayarlar menüsü gösterilemektedir. Ayarlar menüsündeki seçenekler ve işlevleri aşağıda açıklanmıştır:



Şekil 8.20 İşlem Sunucusu Ayarlar Menüsü

Ekranı Temizle: Sunucunun çalışması ile ilgili bilgiler sunucu ekranında gösterilebilir. Bu seçenek sunucu ekranının temizlenmesi için kullanılır.

Mesajları İzle: Sunucunun çalışması ile ilgili bilgiler, hata mesajları, sunucunun gönderdiği ve sunucuya gelen mesajlar sunucu ekranında gösterilebilmektedir. Fakat bu gösterim seçimlidir. Ayarlar menüsünden Mesajları İzle seçilirse sunucu bilgilerinin ekranda gösterilimi sağlanır.

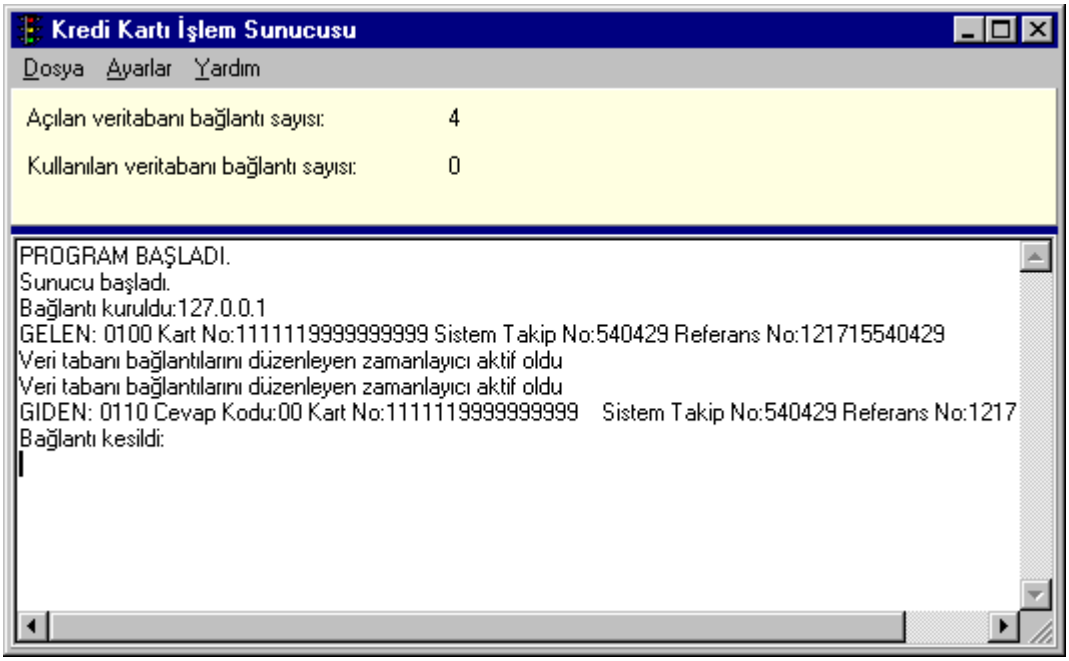
Mesajları Durdur: Seçildiğinde sunucu bilgileri ekranda gösterilmez.

Hareket Kayıt Dosyasını Aç: Seçildiğinde sunucu bilgileri hareket dosyasına yazılır.

Hareket Kayıt Dosyasını Kapat: Seçildiğinde sunucu bilgileri hareket dosyasına yazılmaz.

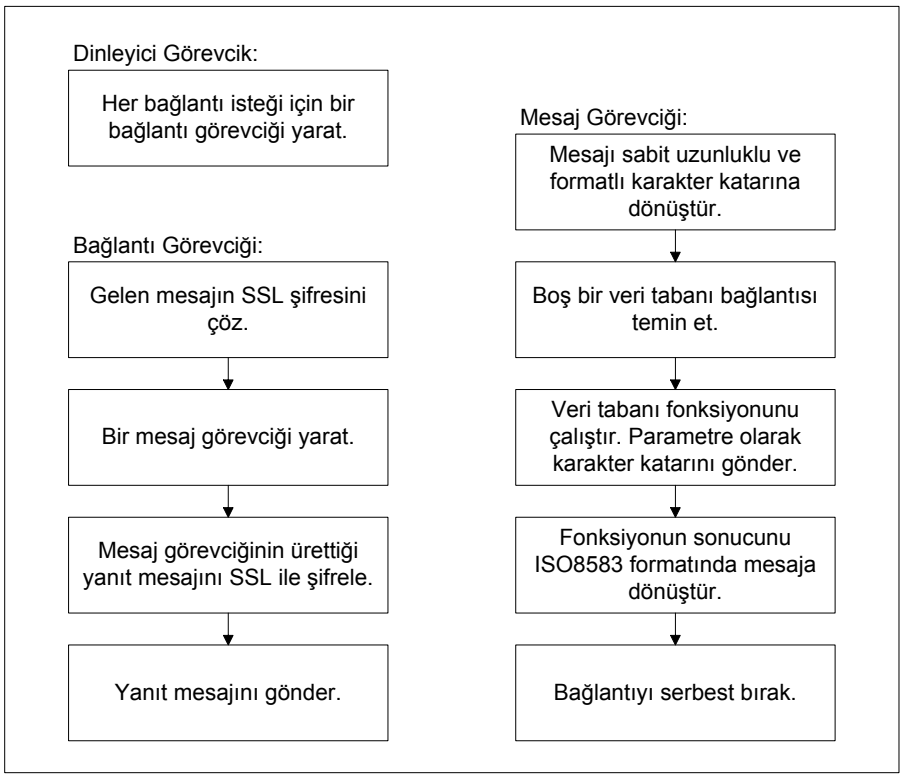
8.2.3. İşlem sunucusunun çalışma mantığı

Şekil 8.21'de işlem sunucunun çalışması gösterilmektedir.



Şekil 8.21 İşlem sunucusunun çalışması

İşlem sunucusunun çalışma mantığı Şekil 8.22'de gösterilen algoritma ile açıklanmıştır.



Şekil 8.22 İşlem sunucusu çalışma mantığı

İşlem sunucusu Delphi programlama dilinde gerçeklenmiştir. Haberleşme işlemleri için Indy'nin sunucu bileşenleri kullanılmıştır. Veri tabanı erişimi, ODBC sürücüsünün OLE DB sağlayıcısı kullanılarak ADO yöntemi ile gerçeklenmiştir.

8.2.4. Veri tabanı bağlantılarının dinamik olarak ayarlanması

İşlem sunucusu başlatıldığında veri tabanına başlangıç veri tabanı bağlantı sayısı kadar bağlantı açar. İşlem sunucusuna gelen her istek için veri tabanı bağlantısı temin edilirken veya istek sonlandığında veri tabanı bağlantısı bırakılırken açılan veri tabanı bağlantı sayısı ve kullanılan veri tabanı bağlantı sayısı kontrol edilir.

Kullanılan veri tabanı bağlantı sayısı, açılan veri tabanı bağlantı sayısının maksimum kullanım yüzdesine ulaşmışsa belli bir sayıda yeni veri tabanı bağlantısı açılır. Kullanılan veri tabanı bağlantı sayısı, açılan veri tabanı bağlantı sayısının minimum kullanım yüzdesine ulaşmışsa belli bir sayıda veri tabanı bağlantısı yok edilir. Bunu algoritma olarak ifade edersek;

Kullanılan veri tabanı bağ. sayısı >= (Açılan veri tabanı bağ. sayısı * MAKY) ise

BDS kadar yeni veri tabanı bağlantısı açılır.

Kullanılan veri tabanı bağ. sayısı <= (Açılan veri tabanı bağ. sayısı * MNKY) ise

BDS kadar veri tabanı bağlantısı yok edilir.

Açılan ve kullanılan veri tabanı bağlantı sayıları anlık olarak sunucu ekranında gösterilir.

8.3. Kredi Kartı Veri Tabanı ve Üzerinde Çalışan Programlar

Veri tabanında kart ve müşteri bilgilerinin, parametre tanımlarının ve işlem detaylarının tutulduğu tablolar bulunmaktadır. Veri tabanı üzerinde çalışan fonksiyonlarla veri tabanını sorgulama ve güncelleme işlemleri yapılmaktadır. Bu fonksiyonlar işlem isteği geldiğinde işlem sunucusu tarafından çalıştırılırlar. Fonksiyonlar veri tabanından yaptıkları sorgulamalarla işlemin onaylanıp onaylanmayacağına karar verirler ve işlem onaylanmışsa veri tabanında gerekli güncellemeleri yaparlar. İşlem onaylanmamışsa da gelen işlem isteği veri tabanına kayıt edilir fakat kart ve müşteri bilgileri güncellenmez. Veri tabanı olarak Oracle kullanılmıştır.

8.3.1. İşlem isteği mesajları

İşlem isteği (ISO8583 formatındaki mesaj tipi 100) ve işlem isteği tekrar (ISO8583 formatındaki mesaj tipi 101) mesajları için kullanılan veri tabanı fonksiyonu veri tabanı üzerinde aşağıdaki kontrol ve güncellemeleri yapar:

Gelen mesajın detayları veri tabanına kayıt edilir.

Gelen işlemin tipine (satış/nakit çekim) karar verilir.

İşlem kuru geçerliliği kontrol edilir.

İşlem isteğinde bulunan kart ve kartın ait olduğu müşteri bilgileri veri tabanında mevcut mu kontrolü yapılır.

Kartın durumu (aktif/iptal) kontrol edilir.

Kartın son kullanım tarihi kontrol edilir.

İşlemin yapıldığı üye işyerinin kategori kodu kontrol edilir.

Kartın kullanılabilir limit bilgileri kontrol edilir.

Müşterinin kullanılabilir limit bilgileri kontrol edilir.

İşlem nakit çekim işlemi ise kartın ve müşterinin kullanılabilir nakit çekim limit ve kalan nakit işlem sayıları kontrol edilir.

Sistemin genel işlem limitleri kontrol edilir.

İşlem tüm kontrollerden geçmişse onaylanmış demektir. Onaylanan işlem için veri tabanına yeni işlem kaydı yaratılır. Kart ve müşterinin kullanılabilir limit bilgileri güncellenir. (İşlem isteği tekrar mesajları için işlemin daha önce veri tabanına kayıt edilip edilmediği kontrol edilir. İşlem daha önceden kayıt edilmiş ise yeni işlem kaydı yaratılmaz, kart ve müşteri bilgileri tekrar güncellenmez.)

Onaylanan veya red edilen işlem için yanıt mesajı oluşturulur. Yanıt mesajının detayları veri tabanına kayıt edilir.

8.3.2. Tavsiye mesajları

Pratikteki kredi kartı işlem akışlarında işlemin yapıldığı kaynak ve kartın ait olduğu finansal kurum arasında çeşitli sistemler ve finansal kuruluşlar mevcuttur. Bazı işlemler için kartın ait olduğu finansal kuruma ulaşılamıyorsa veya işlem tutarı belli limitlerin altında ise aradaki sistemler veya finansal kuruluşlar işleme onay verebilirler. Bu durumda işlemin onaylandığı bilgisi kartın ait olduğu finansal kuruma tavsiye mesajı (ISO8583 formatındaki 120 mesajı) olarak gönderilir. Kartın ait olduğu finansal kurumdan yanıt alınana kadar tavsiye mesajı tekrarlanır (ISO8583 formatındaki 121 mesajı). Tavsiye mesajlarında işlem zaten onaylanmış olduğundan işlem isteği mesajlarındaki kontroller uygulanmaz. Bu mesajlar için kullanılan veri tabanı fonksiyonu aşağıdaki kontrol ve güncellemeleri yapar:

Gelen mesajın detayları veri tabanına kayıt edilir.

İşlemin tipine (satış/nakit çekim/ödeme) karar verilir.

İşlem kurunun geçerliliği kontrol edilir.

İşlem isteğinde bulunan kart ve kartın ait olduğu müşteri bilgileri veri tabanında mevcut mu kontrolü yapılır.

Tavsiye istekleri için veri tabanına yeni işlem kaydı yaratılır. Kart ve müşterinin kullanılabilir limit bilgileri güncellenir. (Tavsiye isteği tekrar mesajları için işlemin daha önce veri tabanına kayıt edilip edilmediği kontrol edilir. İşlem daha önceden kayıt edilmiş ise yeni işlem kaydı yaratılmaz, kart ve müşteri bilgileri tekrar güncellenmez.)

Tavsiye isteği için yanıt mesajı oluşturulur. Yanıt mesajının detayları veri tabanına kayıt edilir.

8.3.3. İptal mesajları

Daha önceden gönderilmiş olan işlemlerin iptal edilmesi için kullanılır. Bu mesajlar için kullanılan veri tabanı fonksiyonu aşağıdaki kontrol ve güncellemeleri yapar:

Gelen mesajın detayları veri tabanına kayıt edilir.

İşlemin tipine (satış/nakit çekim/ödeme) karar verilir.

İşlem kurunun geçerliliği kontrol edilir.

İşlem isteğinde bulunan kart ve kartın ait olduğu müşteri bilgileri veri tabanında mevcut mu kontrolü yapılır.

İptal edilmek istenen orijinal işlem veri tabanında mevcut mu kontrolü yapılır. Mevcut değilse işlem red edilir. Mevcut ise orijinal işlem iptal edilir ve iptal isteği işlem tablosuna kayıt edilir. Kart ve müşterinin kullanılabilir limit bilgileri güncellenir. (İptal isteği tekrar mesajları için orijinal işlemin daha önce iptal edilip edilmediği kontrol edilir. İşlem daha önceden iptal edilmiş ise işlem tablosuna yeni kayıt yaratılmaz, kart ve müşteri bilgileri tekrar güncellenmez.)

İptal isteği için yanıt mesajı oluşturulur. Yanıt mesajının detayları veri tabanına kayıt edilir.

8.4. Sorgulama ve Güncelleme Ekranları

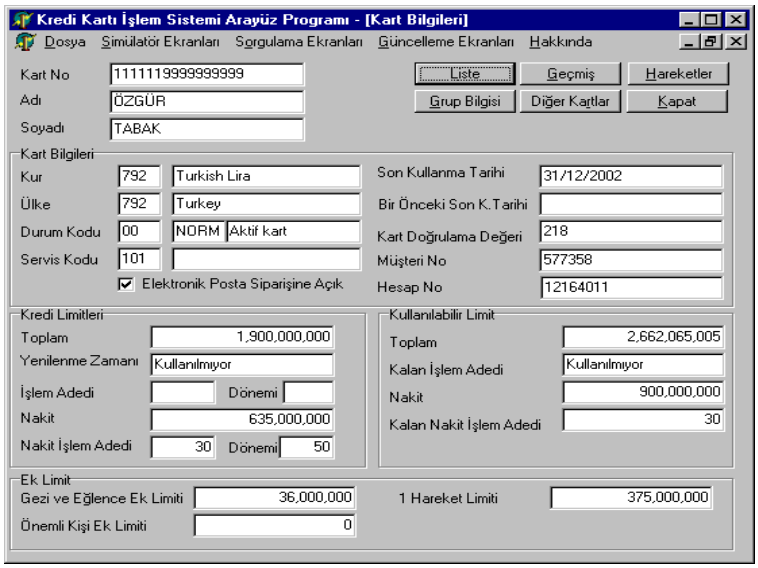
Kredi kartı işlemlerinin ve kart bilgilerinin sorgulandığı ve güncellendiği modüldür. Delphi programla dilinde gerçeklenmiştir, veri tabanı bağlantısı için ADO kullanılmıştır.

8.4.1. Sorgulama ekranları

Kredi kartı işlemlerinin ve kart bilgilerinin sorgulandığı modüldür.

8.4.1.1. Kart bilgileri ekranı

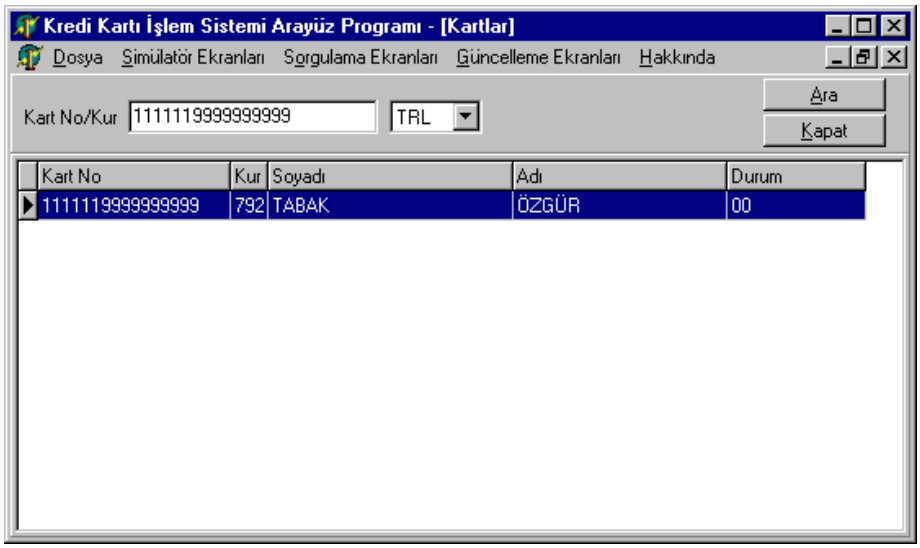
Şekil 8.23'teki Kart Bilgileri ekranında, girilen kart numarasına ait kart, müşteri ve limit bilgileri gösterilmektedir.



Şekil 8.23 Kart Bilgileri Ekranı

8.4.1.2. Kartlar ekranı

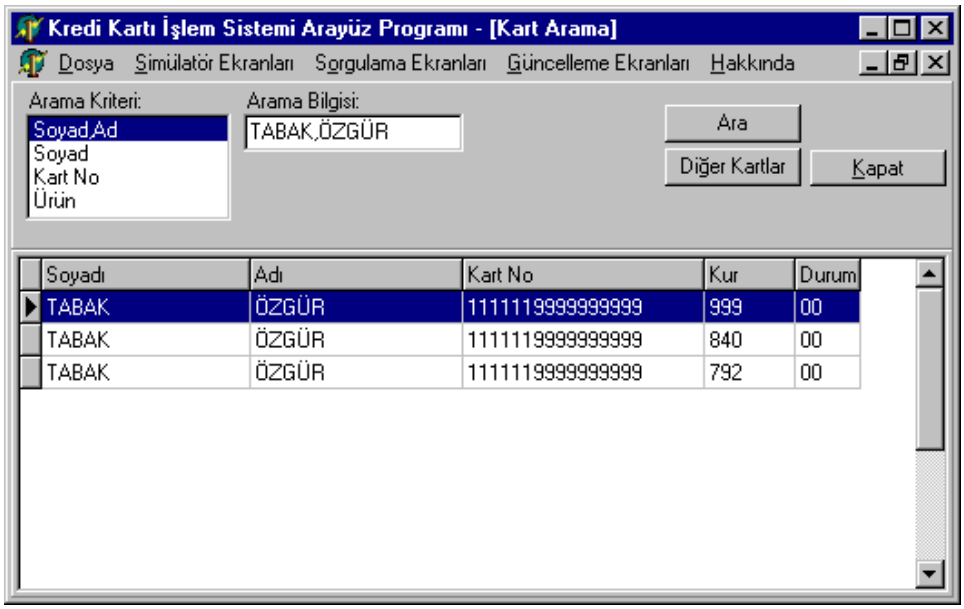
Şekil 8.24'teki Kartlar ekranında, girilen kart numarasının bağlı olduğu müşteriye ait tüm kartlar listelenmektedir.



Şekil 8.24 Kartlar Ekranı

8.4.1.3. Kart arama ekranı

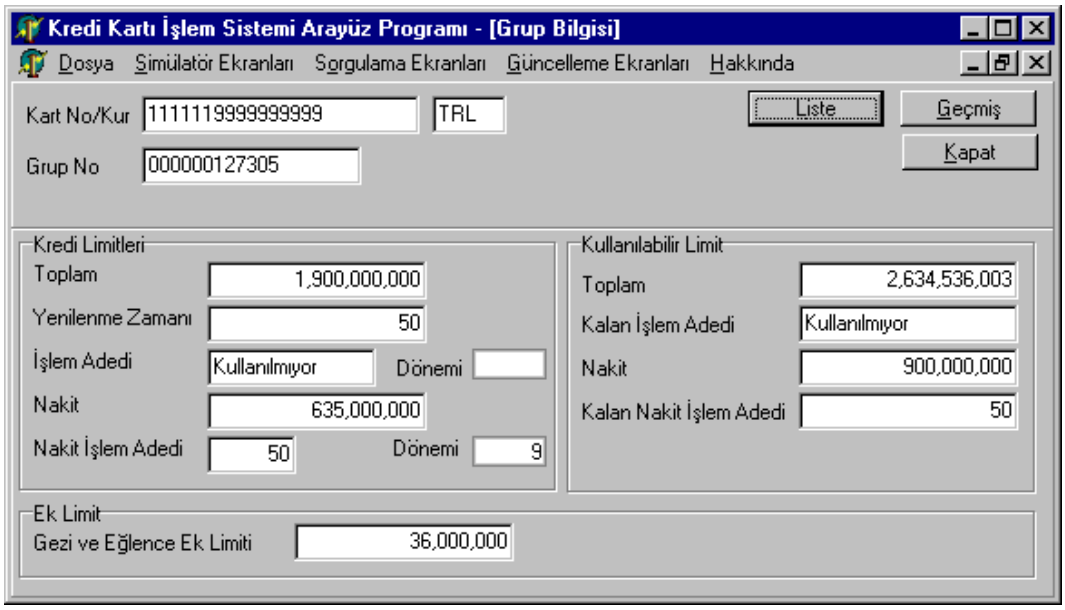
Şekil 8.25'teki Kart Arama ekranında, çeşitli kriterlere göre kart bilgileri sorgulanmakta ve getirilen bilgiler listelenmektedir.



Şekil 8.25 Kart Arama Ekranı

8.4.1.4. Grup bilgisi ekranı

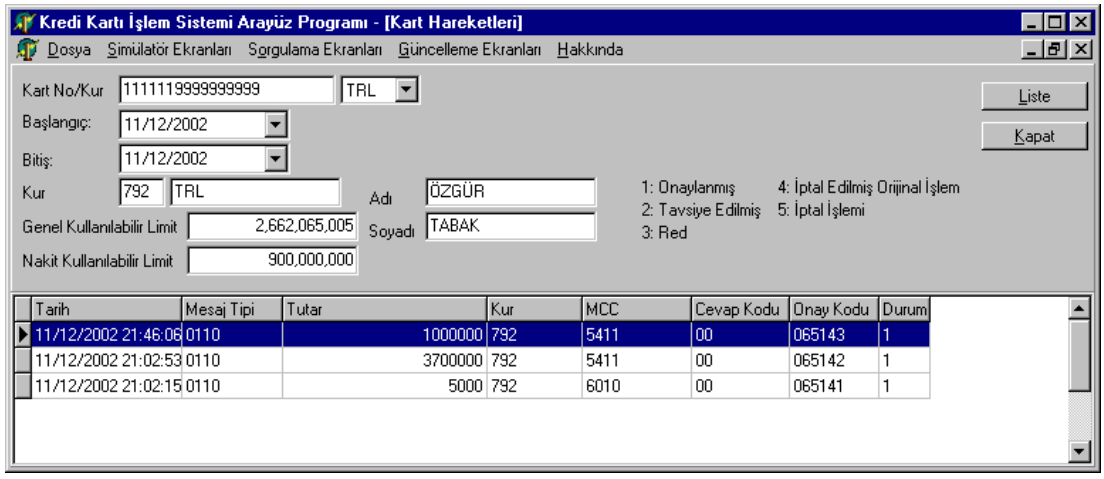
Şekil 8.26'daki Grup Bilgisi ekranında, girilen kart numarasının bağlı olduğu müşteri limit bilgileri gösterilmektedir.



Şekil 8.26 Grup Bilgisi Ekranı

8.4.1.5. Kart hareketleri ekranı

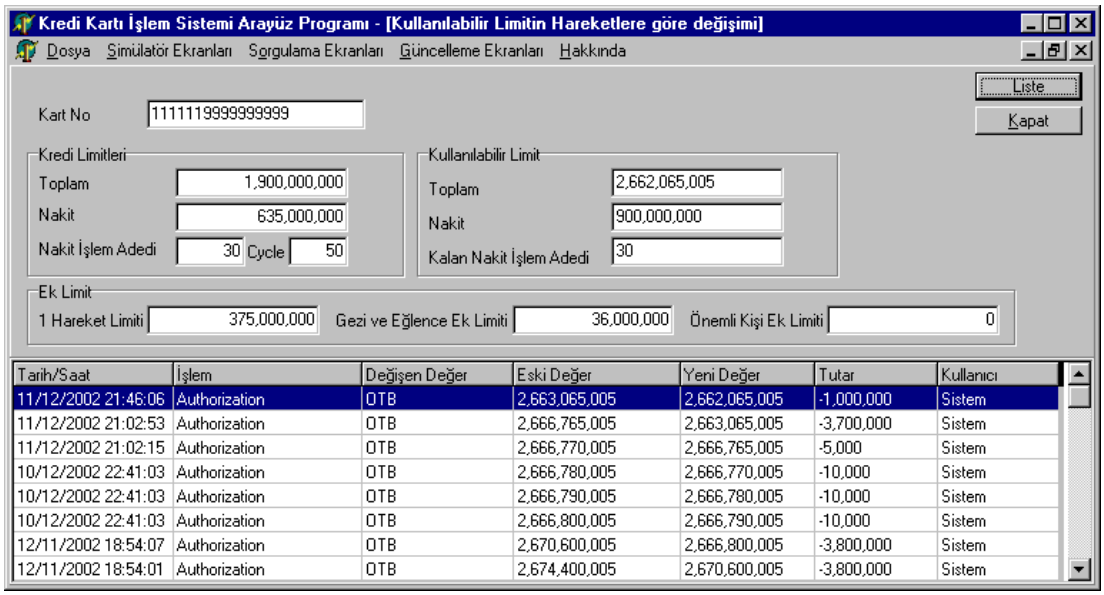
Şekil 8.27'deki Kart Hareketleri ekranında, girilen kart numarasının seçilen tarih aralığında yaptığı işlemler gösterilmektedir.



Şekil 8.27 Kart Hareketleri Ekranı

8.4.1.6. Kullanılabilir limitin hareketlere göre değişimi ekranı

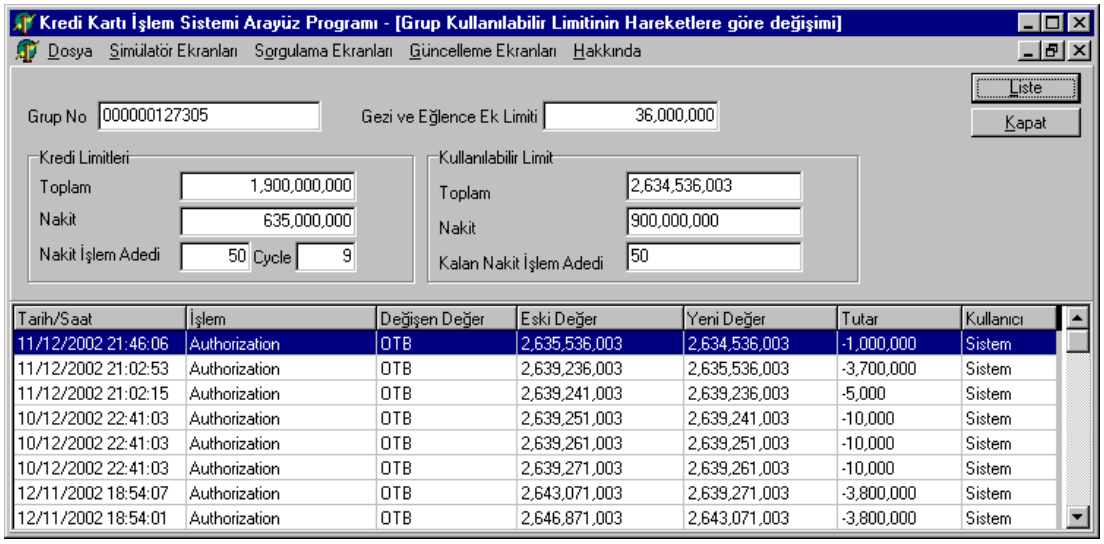
Şekil 8.28'deki Kullanılabilir Limitin Hareketlere Göre Değişimi ekranında, girilen kart numarasının kullanılabilir limit değişim bilgileri gösterilmektedir.



Şekil 8.28 Kullanılabilir Limitin Hareketlere Göre Değişimi Ekranı

8.4.1.7. Grup kullanılabilir limitinin hareketlere göre değişimi ekranı

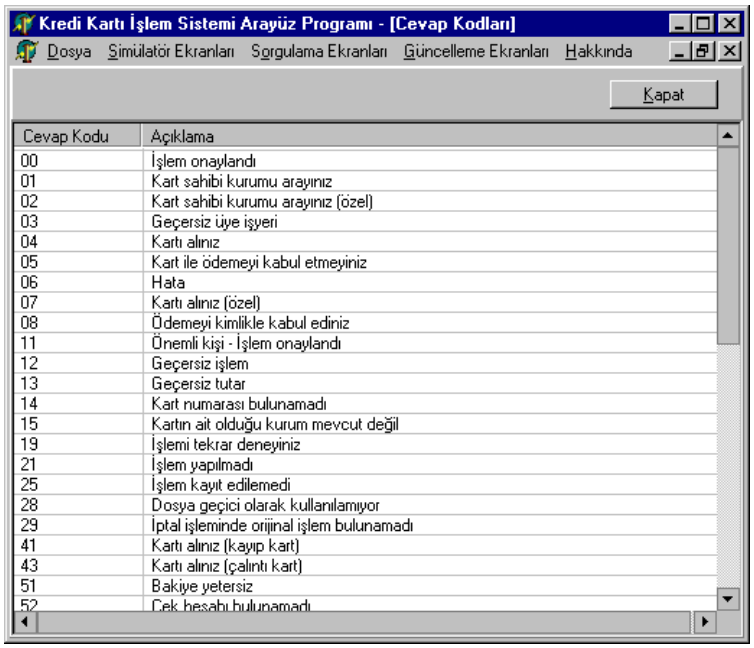
Şekil 8.29'daki Grup Kullanılabilir Limitinin Hareketlere Göre Değişimi ekranında, girilen müşteri numarasının kullanılabilir limit değişim bilgileri gösterilmektedir.



Şekil 8.29 Grup Kullanılabilir Limitinin Hareketlere Göre Değişimi Ekranı

8.4.1.8. Yanıt kodları ekranı

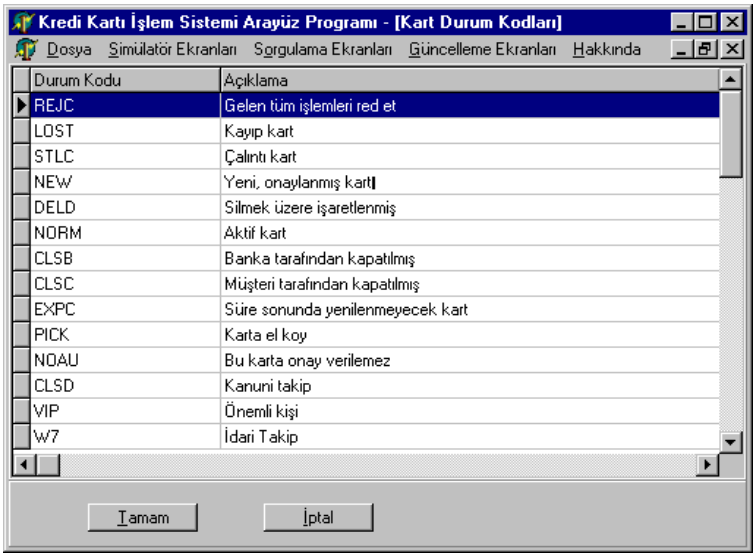
Şekil 8.30'daki Yanıt Kodları ekranında, ISO8583 mesaj formatında kullanılan yanıt kodları ve açıklamaları listelenmektedir.



Şekil 8.30 Yanıt Kodları Ekranı

8.4.1.9. Kart durum kodları ekranı

Şekil 8.31'deki Kart Durum Kodları ekranında, kart durum kodları ve açıklamaları listelenmektedir.



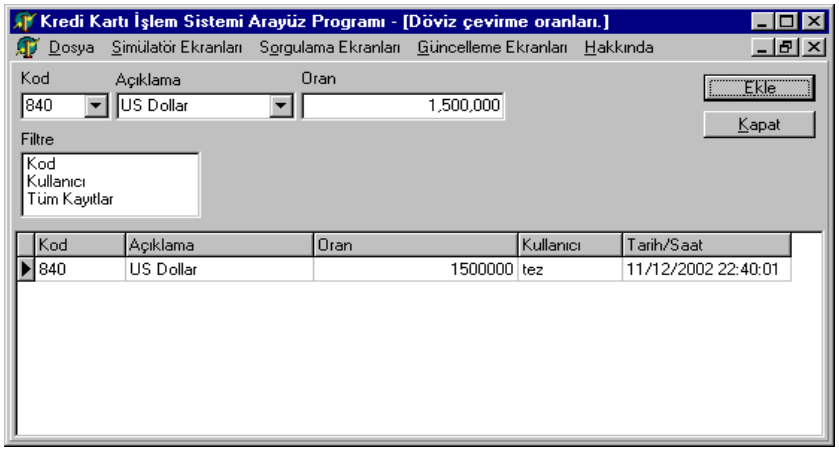
Şekil 8.31 Kart Durum Kodları Ekranı

8.4.2. Güncelleme ekranları

Bilgi güncellemenin yapıldığı modüldür.

8.4.2.1. Döviz çevirme oranları ekranı

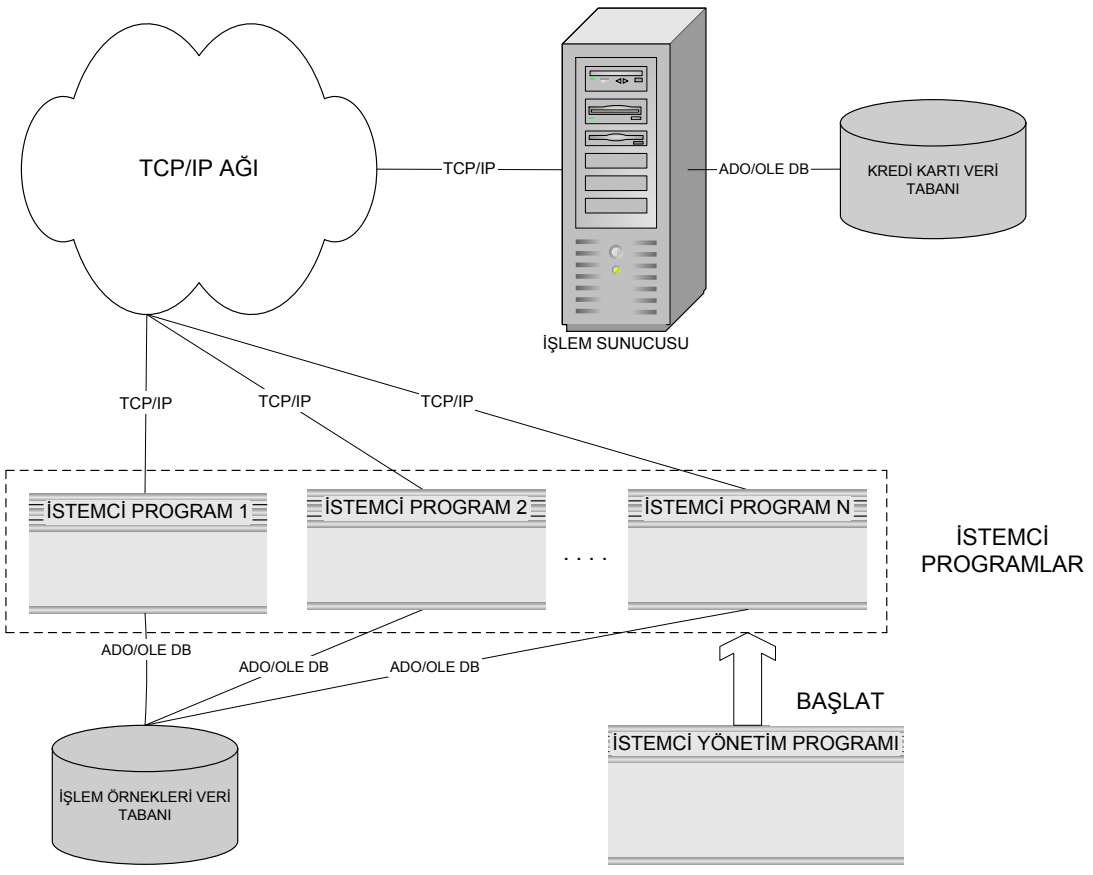
Şekil 8.32'deki Döviz Çevirme Oranları ekranında, çeşitli döviz kurlarından Türk Lirasına çevirme oranları gösterilmektedir.



Şekil 8.32 Döviz Çevirme Oranları Ekranı

9. PERFORMANS ANALİZİ

Bu bölümde kredi kartı işlem sisteminin performansı ve çeşitli koşullardaki davranış biçimi, sistem üzerinde yapılan çeşitli ölçümlere ve bu ölçümler sonucunda elde edilen grafiklere dayanılarak sunulmaktadır. Ölçümleri yapabilmek için, işlem örneklerinin tutulduğu veri tabanından kayıt okuyup işlem sunucusuna yönlendiren İstemci Program ve istemci programların başlatılmasından sorumlu olan İstemci Yönetim Programı tasarlanmıştır. Şekil 9.1'de ölçüm ve grafiklerin elde edilmesinde kullanılan programlar ve aralarındaki ilişkiler gösterilmektedir.



Şekil 9.1 Ölçüm ve Grafiklerin Elde Edilmesinde Kullanılan Programlar

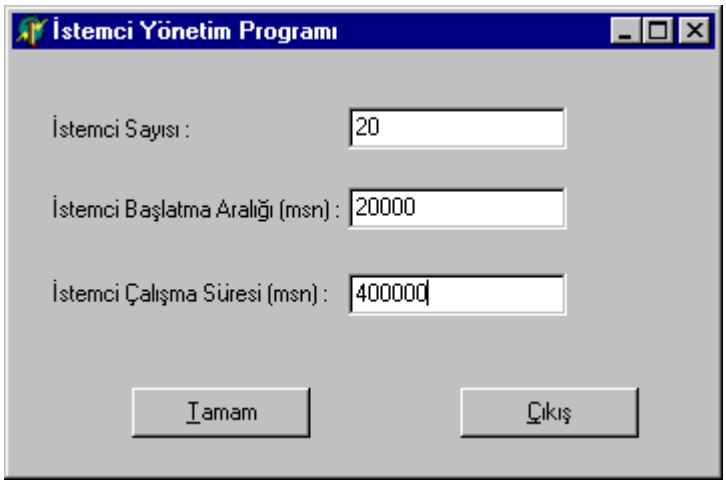
9.1. İstemci Yönetim Programı

Kredi kartı işlem sunucusuna işlem gönderen istemci programların başlatılmasından sorumludur. Şekil 9.2'de gösterilen program arayüzü, istemci programlar ile ilgili çeşitli parametrelerin girişine olanak sağlamaktadır. Bu parametreler :

İstemci sayısı : Başlatılacak olan istemci program sayısını belirlemektedir.

İstemci başlatma aralıkları : İstemci programların ne kadar aralıklarla başlatılacağını belirlemektedir.

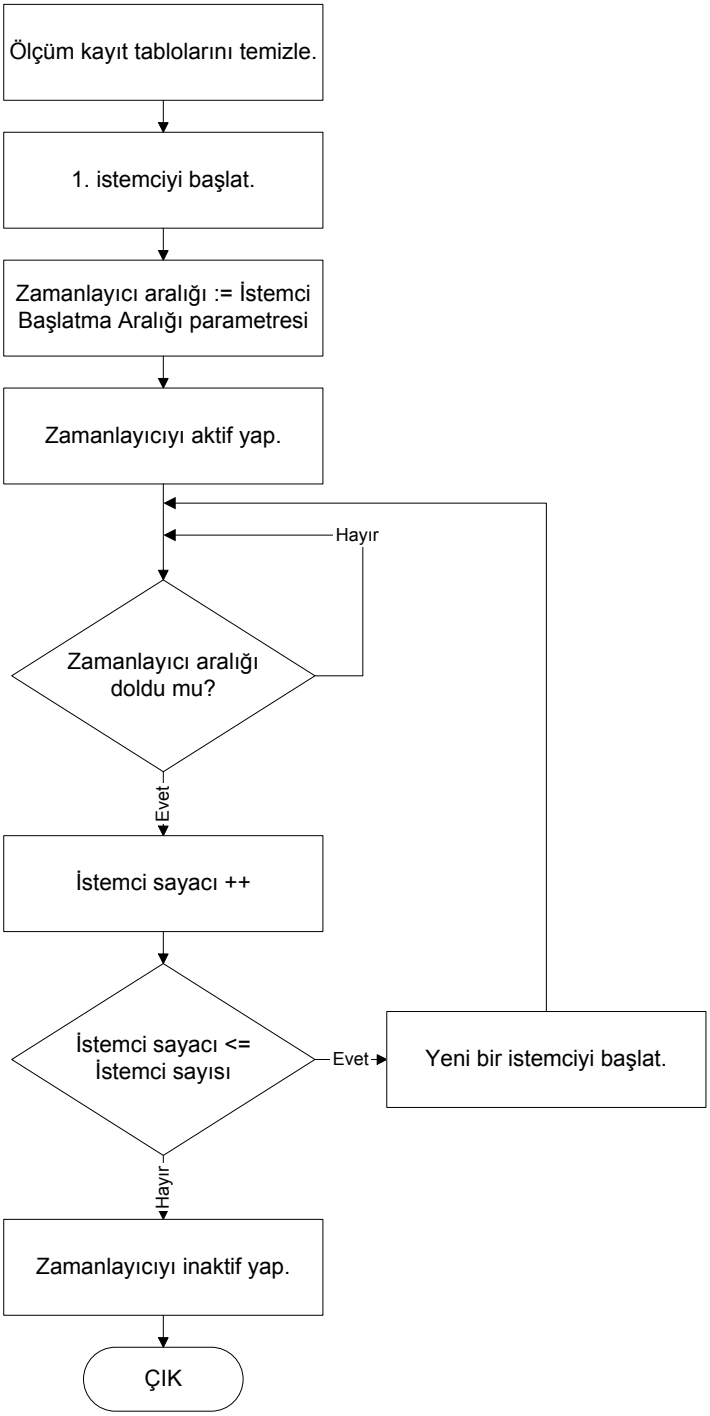
İstemci çalışma süresi : Her bir istemci programının çalışma süresini belirlemektedir.



Şekil 9.2 İstemci Yönetim Programı

Programın çalışma mantığı aşağıda açıklanmıştır:

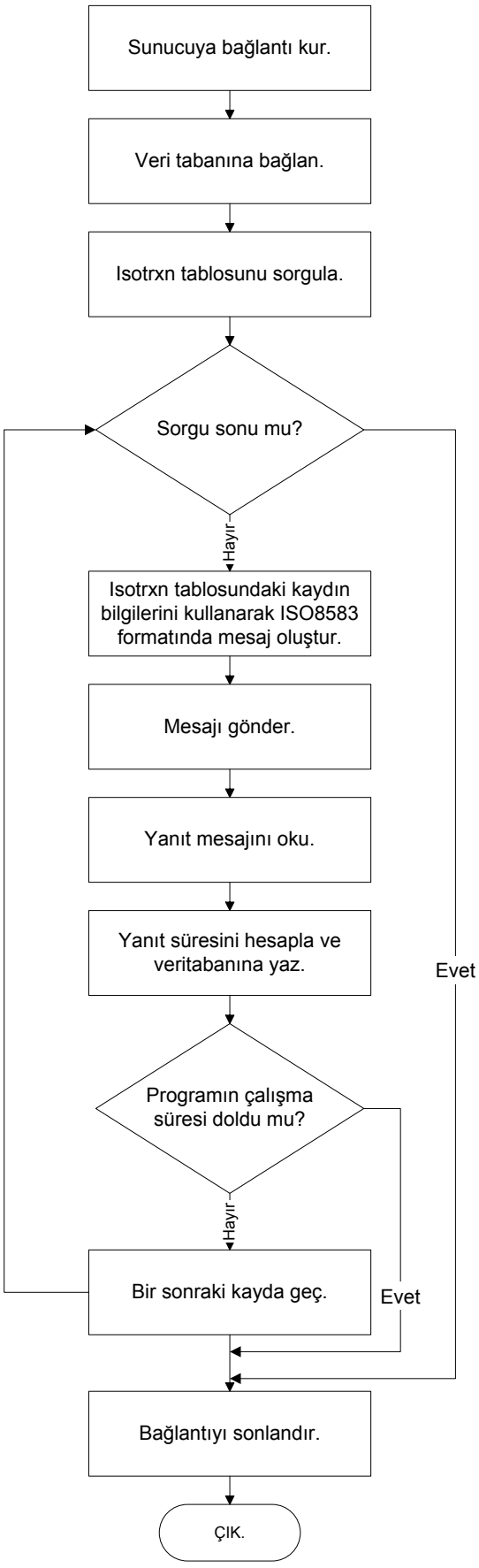
Program üzerinden istemci programlar ile ilgili parametreler girildikten sonra Tamam düğmesine basılır. Tamam düğmesine basılınca çalışan algoritma Şekil 9.3'te gösterilmektedir:



Şekil 9.3 İstemci yönetim programı çalışma mantığı

9.2. İstemci Program

İstemci Yönetim Programı tarafından başlatılan ve veri tabanından okuduğu işlem örneklerini kullanarak işlem sunucusuna işlem gönderen programdır. Programın çalışma mantığı Şekil 9.4'te gösterilmektedir:



Şekil 9.4 İstemci program çalışma mantığı

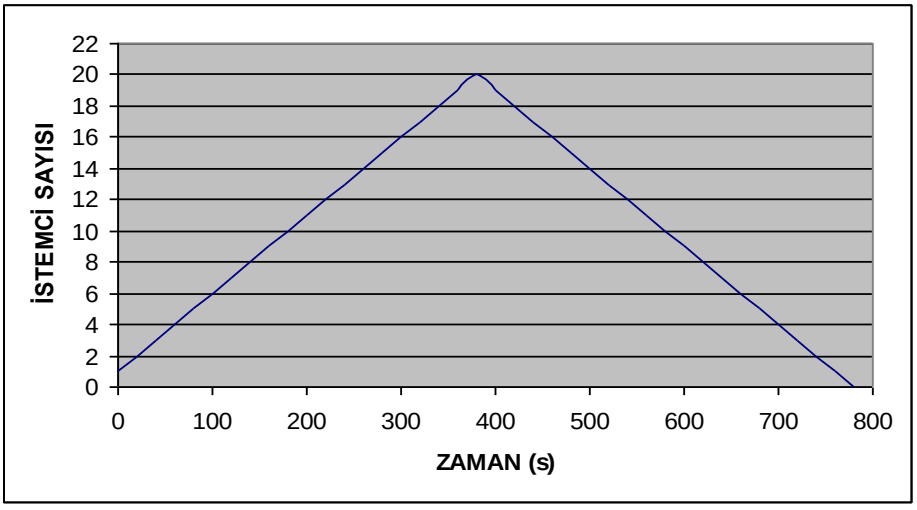
9.3. İşlem Sunucusu ile İlgili Ölçüm ve Grafikler

İşlem sunucusunun performans analizinde kullanılacak verileri elde etmek için Bölüm 9.1'de açıklanan İstemci Yönetim Programı kullanılmıştır. İstemci Yönetim Programını çalıştırmak için kullanılan parametreler ve parametre seçimi ile ilgili açıklamalar aşağıda sunulmuştur:

İstemci program sayısı 20 seçilmiştir çünkü kullanılan veri tabanının konfigürasyonu en fazla 20 istemcinin çalışmasına izin vermektedir.

İstemciler 20 saniye aralıklarla başlatılmıştır. Her bir istemci çalışmaya başladıktan sonra ilk 2-3 saniyeyi veri tabanına bağlantı kurmak ve veri tabanından işlem örneklerini çekmek için harcamaktadır. 20 saniye her bir istemcinin sisteme etkisini gözlemlemek için belirlenen optimum süredir.

İşlem sunucusu üzerindeki yük değişimi, çalışan istemci program sayısına bağlıdır. Yapılan testlerde şöyle bir senaryo uygulanmıştır; işlem sunucusu üzerindeki yük olabilecek en yüksek düzeye ulaşıncaya kadar işlem sunucusu üzerindeki yükleme adım adım arttırılmış, en yüksek düzeye ulaştıktan sonra da işlem sunucusu üzerindeki yükleme adım adım azaltılmıştır. İstemcilerin başlatılma aralıklarının 20 saniye ve istemci sayısının 20 olarak seçildiği düşünülürse bu senaryonun uygulanabilmesi için her bir istemcinin çalışma süresi 400 saniye olmalıdır. Bu senaryoya göre yapılan tüm testler için istemci program sayısının zamana göre değişimi Şekil 9.5'te görüldüğü gibidir.



Şekil 9.5 İstemci Sayısı – Zaman Grafiği

Şekil 9.5'te gösterildiği üzere 0. - 400. saniyeler arasında 20 sn. aralıklarla istemci program sayısı 1 arttırılmış ve 400. – 780. saniyeler arasında istemci program sayısı 1 azaltılmıştır.

Her bir istemcinin veri tabanından, aynı özelliklerde olan fakat birbirlerinden farklı bir işlem grubunu okuması sağlanmıştır. Böylece gönderilen her işlem için veri tabanında çalışan fonksiyonların davranışları aynı olacak ve her bir istemci farklı işlem örneklerini gönderdiği için veri tabanında işlem yürütülürken bekleme ve çakışma durumları oluşmayacaktır.

Yapılan testlerde, işlem sunucusu aşağıdaki yöntemlerle veri tabanına bağlantı kurmaktadır:

- Veri tabanı bağlantı sayısının sabit olduğu yöntem
- Veri tabanı bağlantı sayısının dinamik olarak değiştiği yöntem
- Veri tabanı bağlantılarının her istek için açılıp kapatıldığı yöntem

Yapılan testlerde, aşağıdaki bilgilerin zamana göre değişimleri ölçülmüştür:

- Ortalama yanıt süresi
- Ortalama işlem sayısı
- Açılan ortalama veri tabanı bağlantı sayısı
- Kullanılan ortalama veri tabanı bağlantı sayısı

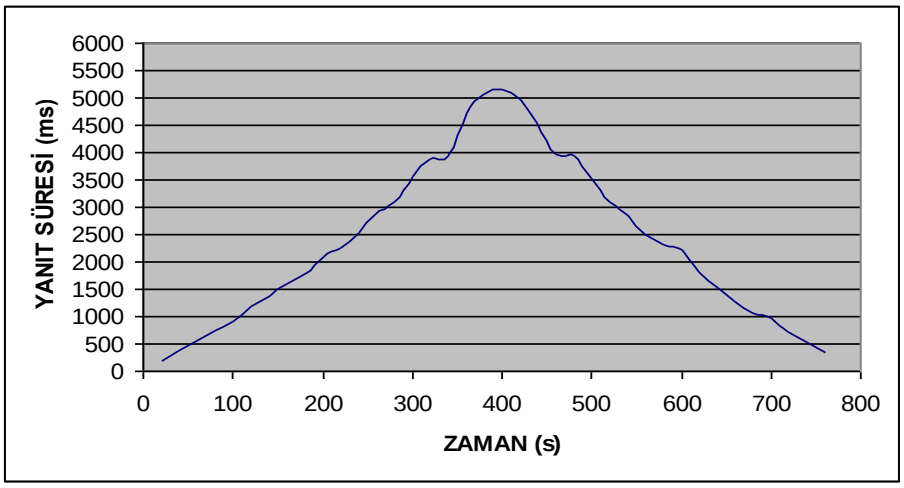
İstemciler 20 saniye aralıklarla başlatıldığı için ortalama ölçümleri 20 saniye aralıklarla yapılmıştır. Böylece istemci program sayılarındaki değişimlerin ortalama değerlere etkisi gözlenmiştir.

Yapılan testler sonucunda elde edilen grafikler ve açıklamaları aşağıda sunulmuştur:

9.3.1. Veri tabanı bağlantı sayısı sabit

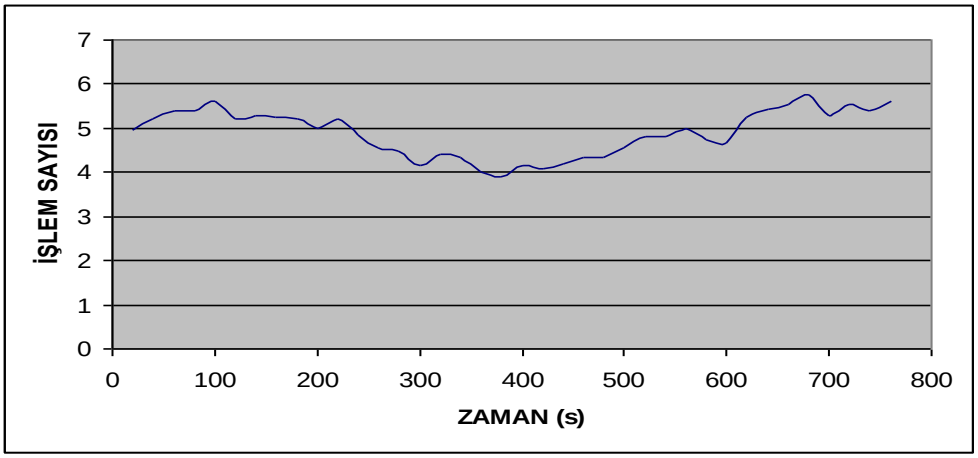
İşlem sunucusu açılırken veri tabanına belirli bir sayıda bağlantı kurar ve kapatılana kadar bağlantı sayısı sabit kalır. Bağlantı sayısı 4 ve 11 seçilerek 2 çeşit ölçüm yapılmıştır.

Bağlantı sayısının 4 seçildiği durumda elde edilen grafikler :



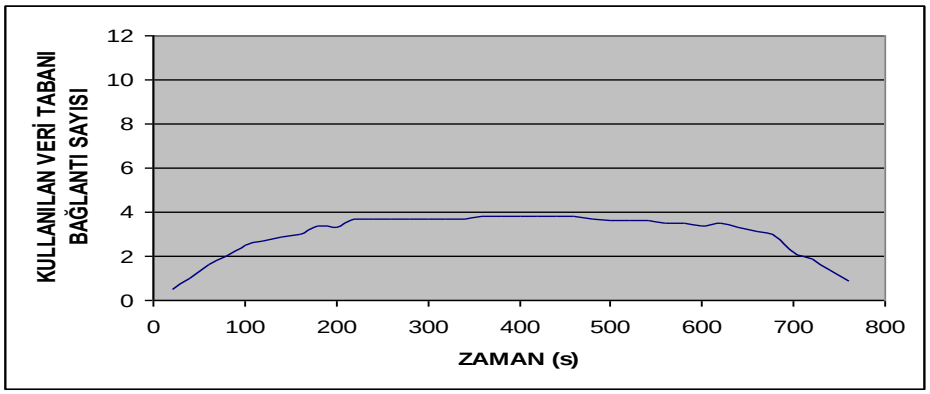
Şekil 9.6 Yanıt Süresi – Zaman Grafiği

Şekil 9.6'da yanıt süresinin zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak yanıt süresinde de doğrusal bir artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak yanıt süresinin de doğrusal olarak azaldığı gözlenmektedir.



Şekil 9.7 İşlem Sayısı – Zaman Grafiği

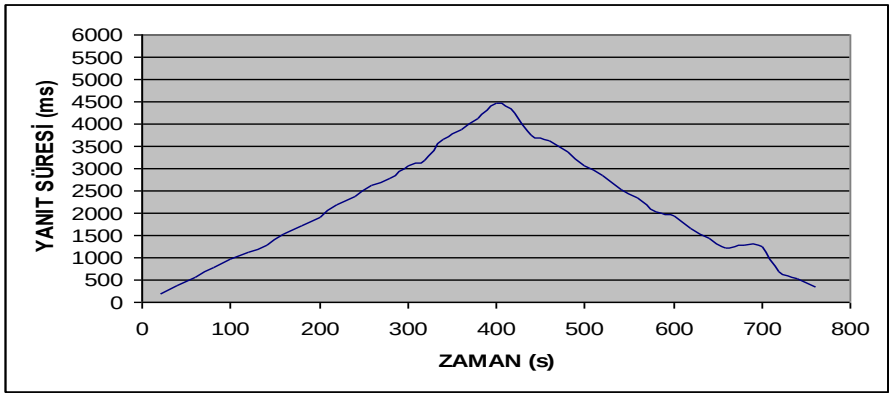
Şekil 9.7'de işlem sayısının zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak saniyede cevap verilen işlem sayısının genel olarak azaldığı gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak saniyede cevap verilen işlem sayısının genel olarak arttığı gözlenmektedir.



Şekil 9.8 Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği

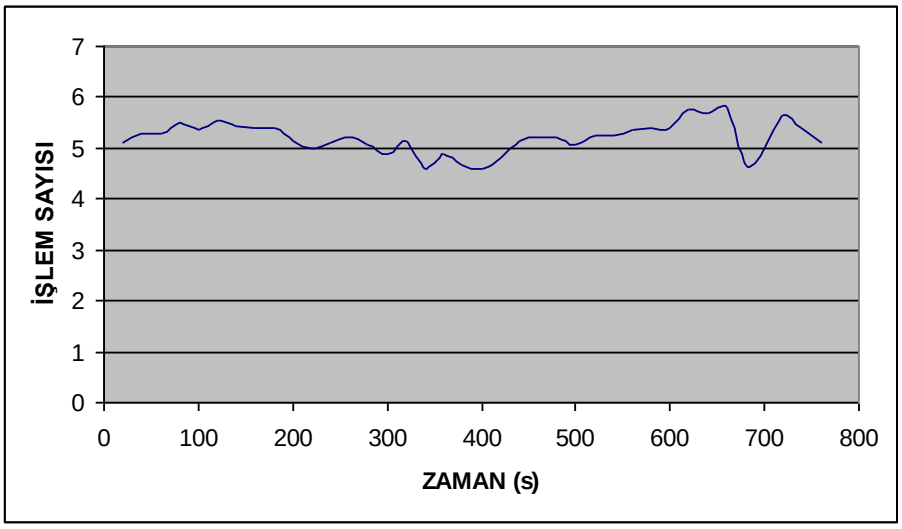
Şekil 9.8'de kullanılan veri tabanı bağlantı sayısının zamana göre değişimi gösterilmektedir. 0. – 200. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak bağlantı sayısında da artış gözlenmektedir. 200. – 600. saniyeler arasındaki bağlantı sayısı yaklaşık olarak sabit ve 4 civarındadır. 600. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak bağlantı sayısının da azaldığı gözlenmektedir.

Bağlantı sayısının 11 seçildiği durumda elde edilen grafikler :



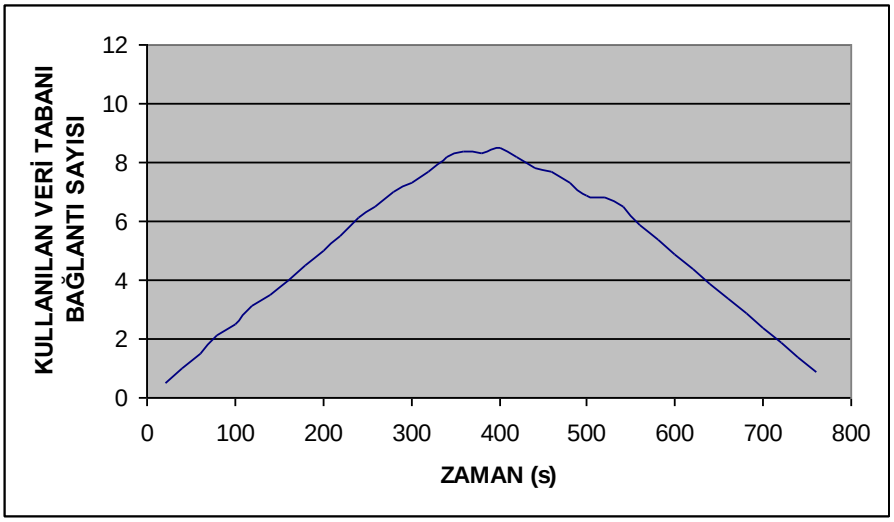
Şekil 9.9 Yanıt Süresi – Zaman Grafiği

Şekil 9.9'da yanıt süresinin zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak yanıt süresinde de doğrusal bir artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak yanıt süresinin de doğrusal olarak azaldığı gözlenmektedir.



Şekil 9.10 İşlem Sayısı – Zaman Grafiği

Şekil 9.10'da işlem sayısının zamana göre değişimi gösterilmektedir. İşlem sayısı genel olarak 5 civarındadır.



Şekil 9.11 Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği

Şekil 9.11'de kullanılan veri tabanı bağlantı sayısının zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak bağlantı sayısında artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak bağlantı sayısının azaldığı gözlenmektedir.

Bağlantı sayısının sabit tutulduğu durumda, bağlantı sayısına bağlı olarak farklı grafikler elde edilmiştir. Bağlantı sayısının 4 ve 11 olduğu durumda elde edilen grafikler karşılaştırılırsa;

Bağlantı sayısının 4 olduğu durumda, işlem istekleri karşılanırken boş bir veri tabanı bağlantısı elde edebilmek için daha uzun süreli beklemeler oluşmaktadır. Bu nedenle yanıt süreleri daha yüksek olup, cevap verilen işlem sayıları daha düşüktür. Açılan veri tabanı bağlantılarının kullanım oranı daha yüksektir, ölçüm yapılan zamanın büyük bir kısmında kullanılan veri tabanı bağlantı sayısı 3.5 ila 4 arasındadır.

Bağlantı sayısının 11 olduğu durumda ise, açılan veri tabanı bağlantılarının kullanım oranı daha düşüktür, işlem sunucusu kullandığından daha fazla kaynağı elinde tutmaktadır. Sadece küçük bir zaman aralığında kullanılan veri tabanı bağlantı sayısı 8'in üzerindedir.

9.3.2. Veri tabanı bağlantı sayısı dinamik

İşlem sunucusu açılırken veri tabanına başlangıç sayısı kadar bağlantı kurar ve anlık olarak kullanılan veri tabanı bağlantı sayısına göre dinamik olarak ayarlamalar yapar. İşlem sunucusu veri tabanı bağlantı sayılarını aşağıdaki algoritmaya göre dinamik olarak belirlemektedir:

Kullanılan veri tabanı bağ. sayısı >= (Açılan veri tabanı bağ. sayısı * MAKY) ise

BDS kadar yeni veri tabanı bağlantısı açılır.

Kullanılan veri tabanı bağ. sayısı <= (Açılan veri tabanı bağ. sayısı * MNKY) ise

BDS kadar veri tabanı bağlantısı yok edilir.

BVBS ve BDS parametrelerinin değerleri aynı kalmak koşulu ile MNKY ve MAKY parametre değerleri değiştirilerek 2 adet ölçüm yapılmıştır.

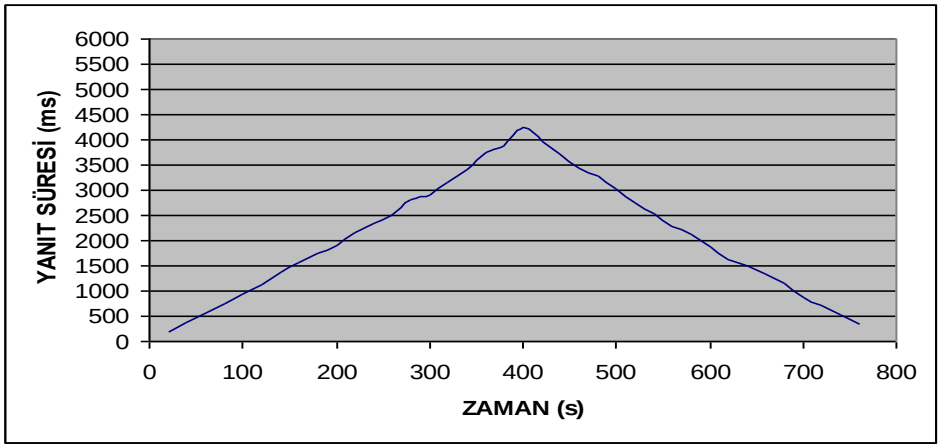
1.ölçüm:

BVBS = 4

MNKY = 10

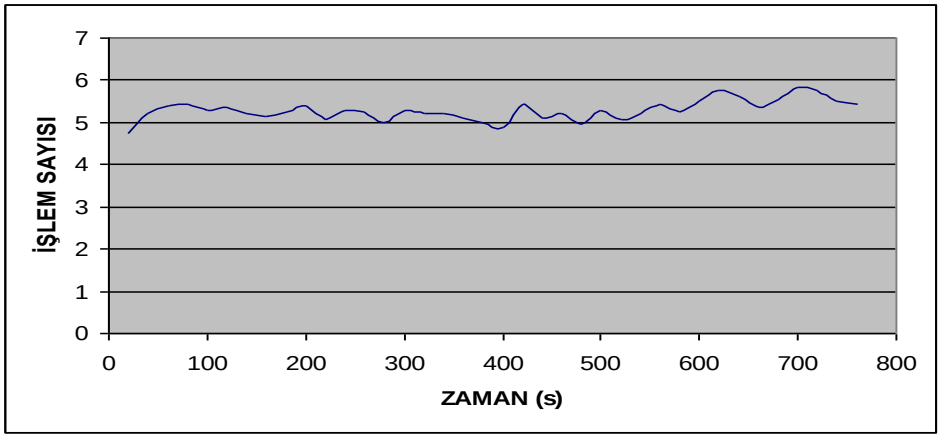
MAKY = 80

BDS = 2



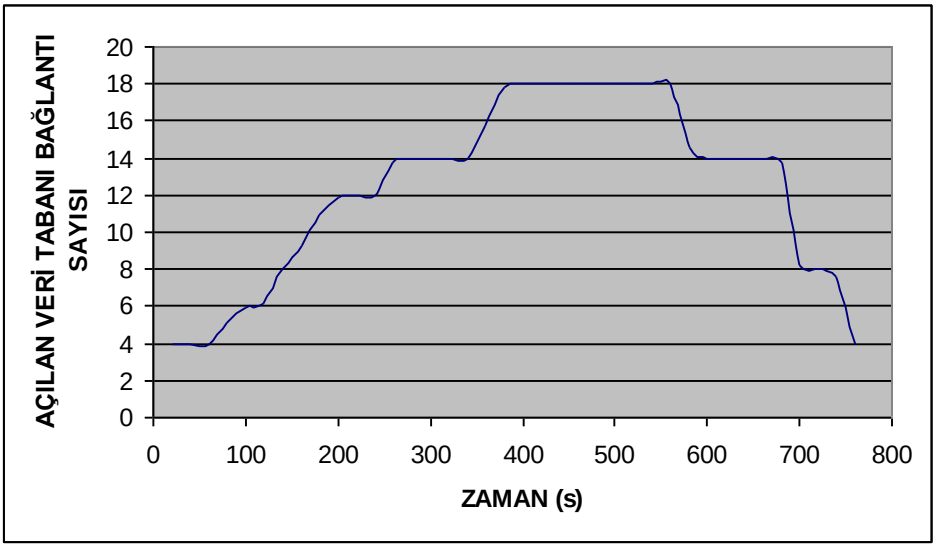
Şekil 9.12 Yanıt Süresi – Zaman Grafiği

Şekil 9.12'de yanıt süresinin zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak yanıt süresinde de doğrusal bir artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak yanıt süresinin de doğrusal olarak azaldığı gözlenmektedir.



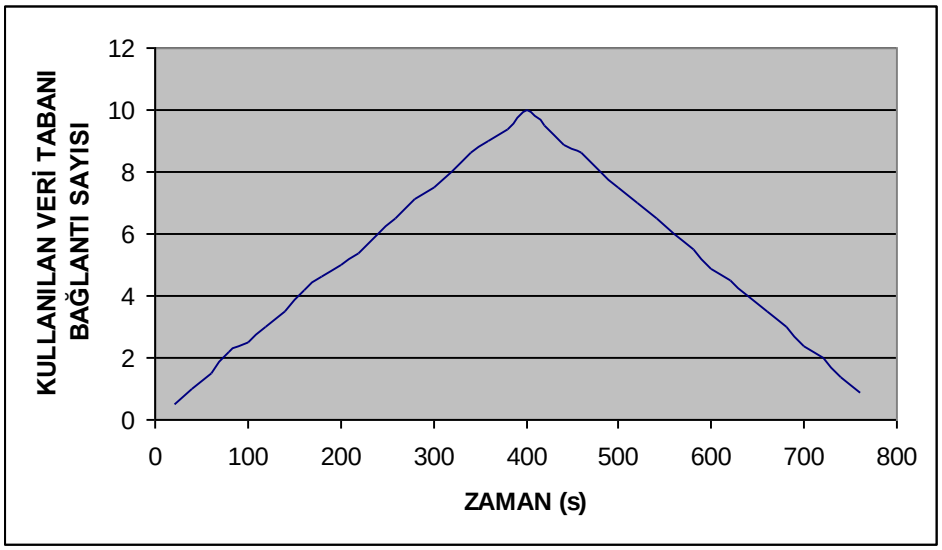
Şekil 9.13 İşlem Sayısı – Zaman Grafiği

Şekil 9.13'te işlem sayısının zamana göre değişimi gösterilmektedir. İşlem sayısı genel olarak 5 civarındadır.



Şekil 9.14 Açılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği

Şekil 9.14'te açılan veri tabanı bağlantı sayısının zamana göre değişimi gösterilmektedir. Grafikteki bağlantı sayıları değişiminin yukarıdaki parametre değerleri ve kullanılan algoritmaya uygun olarak gerçekleştiği gözlenmektedir.



Şekil 9.15 Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği

Şekil 9.15'te kullanılan veri tabanı bağlantı sayısının zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına

bağlı olarak bağlantı sayısında artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak bağlantı sayısının azaldığı gözlenmektedir.

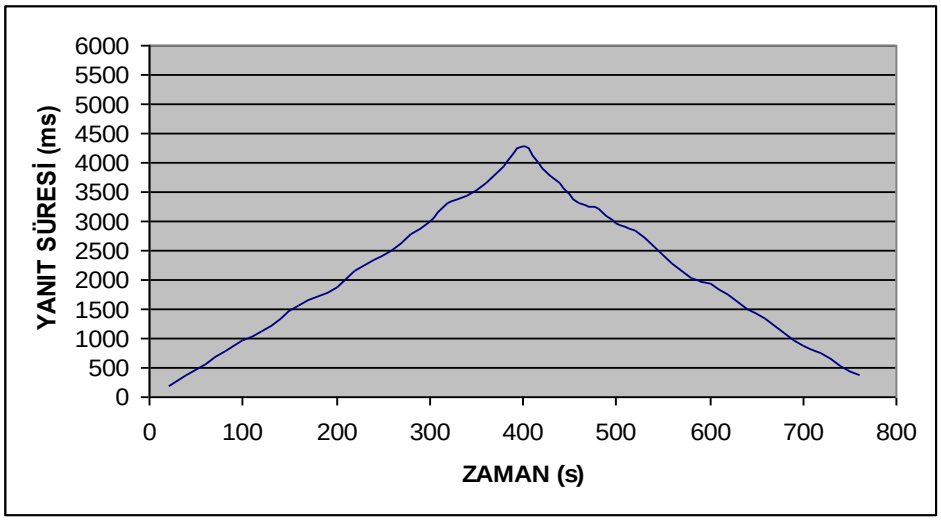
2.ölçüm:

BVBS = 4

MNKY = 50

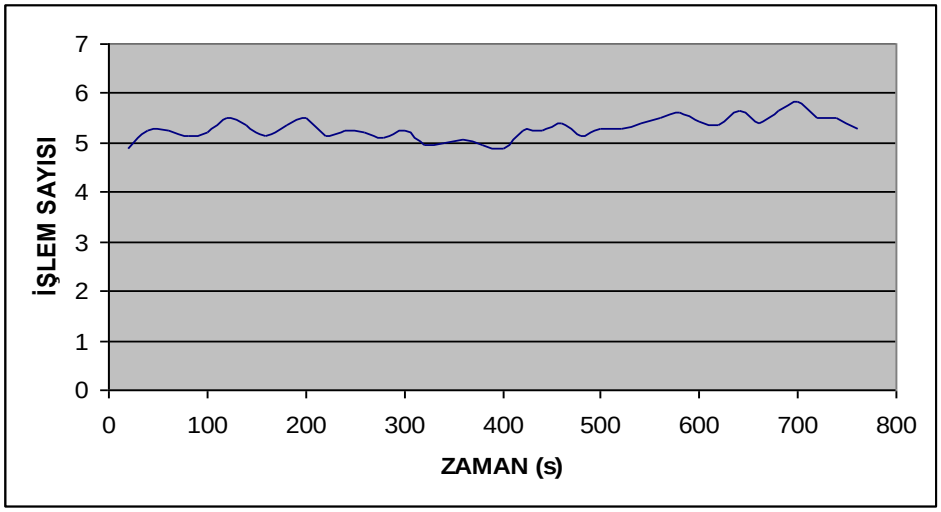
MAKY = 90

BDS = 2



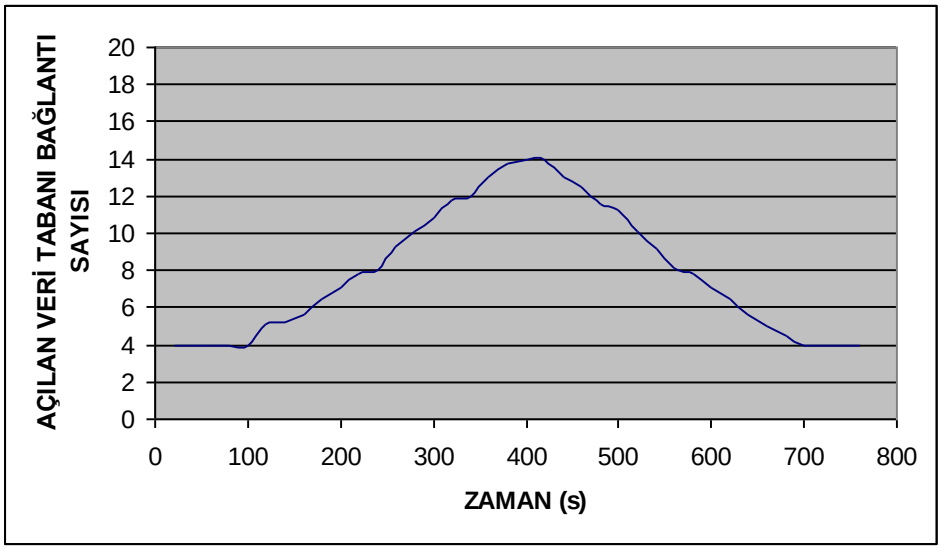
Şekil 9.16 Yanıt Süresi – Zaman Grafiği

Şekil 9.16'da yanıt süresinin zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak yanıt süresinde de doğrusal bir artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak yanıt süresinin de doğrusal olarak azaldığı gözlenmektedir.



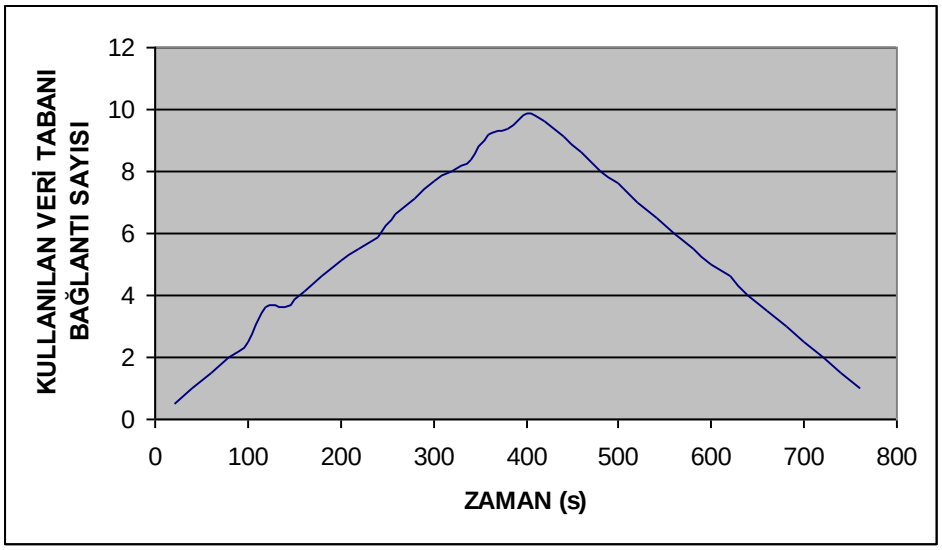
Şekil 9.17 İşlem Sayısı – Zaman Grafiği

Şekil 9.17'de işlem sayısının zamana göre değişimi gösterilmektedir. İşlem sayısı genel olarak 5 civarındadır.



Şekil 9.18 Açılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği

Şekil 9.18'de açılan veri tabanı bağlantı sayısının zamana göre değişimi gösterilmektedir. Grafikteki bağlantı sayıları değişiminin yukarıdaki parametre değerleri ve kullanılan algoritmaya uygun olarak gerçekleştiği gözlenmektedir.



Şekil 9.19 Kullanılan Veri Tabanı Bağlantı Sayısı – Zaman Grafiği

Şekil 9.19'da kullanılan veri tabanı bağlantı sayısının zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak bağlantı sayısında artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak bağlantı sayısının azaldığı gözlenmektedir.

Bağlantı sayısının dinamik olarak belirlendiği durumda, minimum ve maksimum kullanım yüzdesi parametrelerinin değerlerindeki değişimlere bağlı olarak farklı grafikler elde edilmiştir. Yapılan 2 ölçüm karşılaştırılırsa;

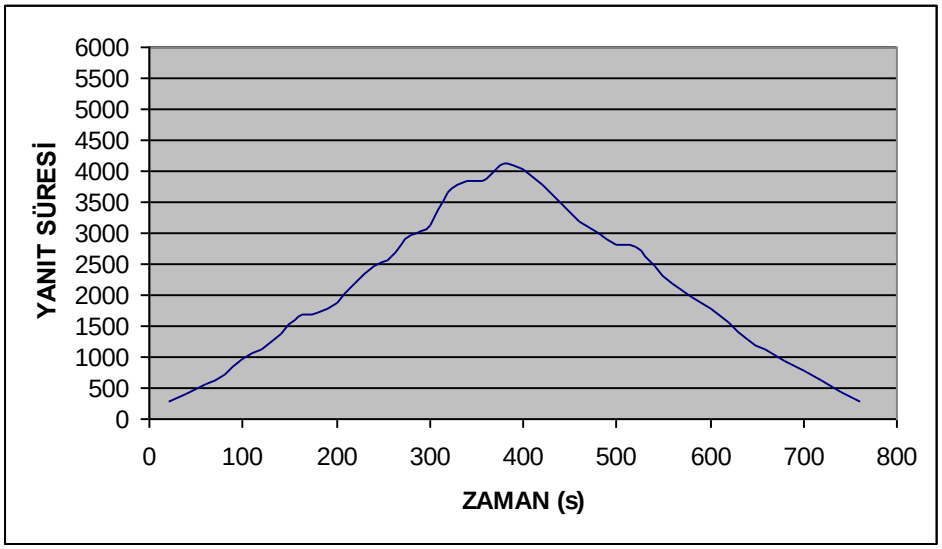
Her iki ölçümde de yanıt süresi – zaman, işlem sayısı – zaman, kullanılan veri tabanı bağlantısı – zaman grafikleri yaklaşık olarak aynı olmakla birlikte açılan veri tabanı bağlantı sayısı – zaman grafikleri farklıdır.

Minimum kullanım yüzdesinin 10 ve maksimum kullanım yüzdesinin 80 olduğu 1. ölçümde, minimum kullanım yüzdesi düşük olduğundan dinamik ayarlamada bağlantı sayısının düşürülmesi olasılığı daha düşüktür. Bu nedenle daha fazla veri tabanı bağlantısı açılmaktadır ve bazı zaman aralıklarında bağlantı sayısı sabit kalmaktadır. Açılan bağlantı sayılarının en fazla 10 tanesi kullanıldığından kullanım oranı daha düşüktür ve işlem sunucusu kullandığından daha fazla kaynağı elinde tutmaktadır.

Minimum kullanım yüzdesinin 50 ve maksimum kullanım yüzdesinin 90 olduğu 2. ölçümde, minimum kullanım yüzdesi daha yüksek olduğundan dinamik ayarlamada bağlantı sayısının düşürülmesi olasılığı daha fazladır. Bu sebeple, açılan veri tabanı bağlantı sayısı, birinci ölçümde elde edilen grafiğin aksine doğrusal bir değişim göstermektedir. İkinci ölçümde kullanım oranı daha yüksektir ve kaynaklar daha verimli kullanılmaktadır.

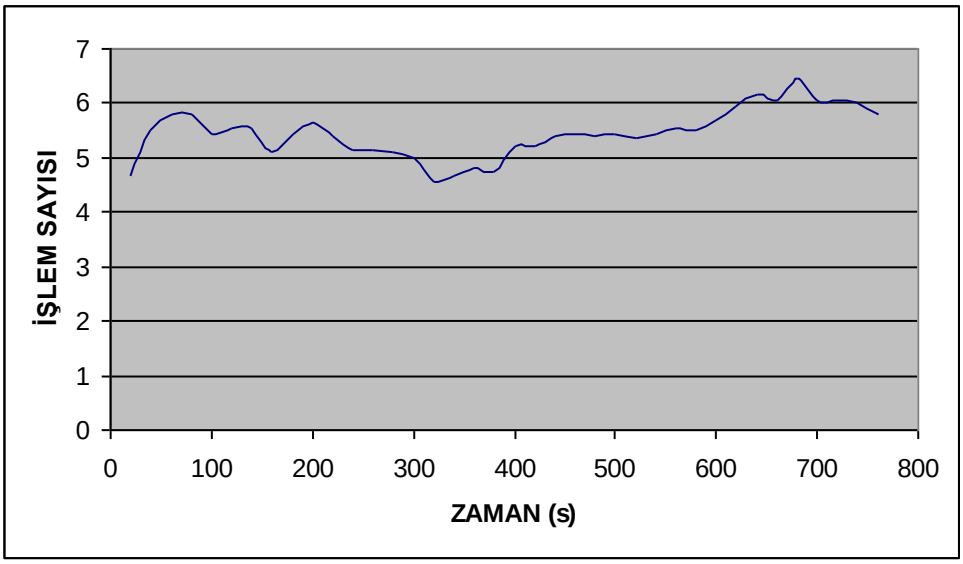
9.3.3. Veri tabanı bağlantılarının her istek için açılıp, kapatıldığı durum

İşlem sunucusu açılırken veri tabanı bağlantısı kurulmaz. Sunucuya işlem isteği geldiğinde veri tabanı bağlantısı kurulur, istek sonlandığında da bağlantı kapatılır. Elde edilen grafikler aşağıda sunulmuştur:



Şekil 9.20 Yanıt Süresi – Zaman Grafiği

Şekil 9.20'de yanıt süresinin zamana göre değişimi gösterilmektedir. 0. – 400. saniyeler arasında zamanla artan istemci program sayısına bağlı olarak yanıt süresinde de doğrusal bir artış gözlenmektedir. 400. – 780. saniyeler arasında da zamanla azalan istemci program sayısına bağlı olarak yanıt süresinin de doğrusal olarak azaldığı gözlenmektedir.



Şekil 9.21 İşlem Sayısı – Zaman Grafiği

Şekil 9.21'de işlem sayısının zamana göre değişimi gösterilmektedir. İşlem sayısı genel olarak 5 ila 6 civarındadır.

9.3.4. Veri tabanı bağlantı kurma yöntemlerinin karşılaştırılması

Veri tabanı bağlantı sayısının sabit tutulduğu yöntemde, aynı yük altında yanıt süreleri ve işlem sayılarının seçilen bağlantı sayısına göre değiştiği gözlenmiştir. Bu yöntemde, sistemin yük durumuna göre açılan veri tabanı bağlantı sayıları dinamik olarak belirlenmediğinden, bazı durumlarda bağlantı sayısı yetersiz kalırken, bazı durumlarda da ihtiyaç duyulandan daha fazla kaynak elde tutulmaktadır. Örneğin, düşük bir sayı seçildiğinde ortalama yanıt süresinin arttığı ve ortalama işlem sayısının azaldığı gözlenmiştir. Seçilecek sayıyı belirlemek için bir takım ölçümler yapılmalı ve optimum sayı belirlenmelidir. Bu yöntemin üstünlüğü, her durumda belli sayıda bağlantının işlem sunucusuna tahsis edilmesini garanti etmesidir.

Veri tabanı bağlantı sayısının dinamik olarak değiştiği yöntemde, aynı yük altında açılan veri tabanı bağlantı sayısı dinamik ayarlamada kullanılan parametrelere göre değişmektedir. Parametrelerin uygun seçilmediği durumlarda ihtiyaç duyulandan daha fazla veri tabanı bağlantısı açılabilmektedir. Fakat, uygun parametrelerin seçildiği durumda açılan veri tabanı bağlantı sayılarının optimuma yakın olduğu gözlenmiştir.

Yukarıdaki iki yöntemde, açılan veri tabanı bağlantıları ortak kullanılan bir kaynak olduğu için, mesaj görevciklerinin bu kaynağa erişimleri semafor yapıları ile denetlenmektedir. Bu sebeple mesaj görevcikleri boş olan bir veri tabanı bağlantısı temin ederken ya da kullandıkları veri tabanı bağlantısını serbest bırakırken birbirlerini bekleyebilmektedirler.

Veri tabanı bağlantılarının her istek için açılıp kapatıldığı yöntemde, ortak bir kaynak kullanımı söz konusu değildir. Bu sebeple semafordan kaynaklanan bekleme süreleri oluşmamaktadır. Buna karşılık her istek için bir tane veri tabanı bağlantısı açılmakta ve yanıt alındıktan sonra kapatılmaktadır. Bu işlem de belli bir süre almaktadır. Fakat yapılan testler esnasında veri tabanı yoğun olarak kullanılmadığı için yanıt süresi ve işlem sayısında diğer iki yönteme göre değişiklik gözlenmemiştir. Bu yöntemin dezavantajı, belli sayıda veri tabanı bağlantısı işlem sunucusuna tahsis edilmediği için veri tabanının yüklü olduğu durumlarda istenilen bağlantının elde edilememe olasılığının olmasıdır ve bu durum işlem sunucusu gibi kritik bir uygulamada risk oluşturmaktadır.

Kullanılacak olan yöntem ve bu yöntemin ayrıntılarına, uygulama ve veri tabanı üzerinde yapılan bir takım ölçümlerin değerlendirilmesi sonucunda karar verilmelidir.

10. SONUÇLAR VE TARTIŞMA

Bu çalışmada, çeşitli kaynaklardan gelen kredi kartı işlem isteklerini karşılayan, işlem kaynağı ve tipine göre gerekli finansal kontrolleri yaparak işlemin onaylanıp onaylanmayacağına karar veren bir sistem tasarlanmıştır.

İşlem kaynakları simülasyon programı üzerinden tanımlanmaktadır. İşlem isteği yaratmak için gereken bilgiler simülasyon ekranlarından girilir ve ISO8583 formatında bir istek mesajı oluşturulur. Simülasyon programı işlem isteğini gönderebilmek için işlem sunucusu ile TCP/IP bağlantısı kurar.

İşlem sunucusu, gelen işlem isteklerini belli bir iskeleden dinler ve bağlantı isteği geldiğinde bir bağlantı görevciği yaratır. Bağlantı görevciği sorumlu olduğu bağlantı üzerinden gelen her işlem isteği için bir mesaj görevciği yaratır. Mesaj görevciği işlemle ilgili finansal kontrolleri yapmak için veri tabanındaki bir programı çalıştırır, veri tabanı programının döndürdüğü bilgileri kullanarak ISO8583 formatındaki yanıt mesajını oluşturur. Bu yanıt mesajı daha önceden kurulan bağlantı üzerinden simülasyon programına gönderilir. Simülasyon programı, gelen yanıt mesajını değerlendirir ve işlem sonucunu ekranda gösterir. Simülasyon programı ve işlem sunucusu arasındaki mesaj güvenliği SSL yöntemi ile sağlanmıştır.

Veri tabanı programları işlem ile ilgili finansal kontrolleri yaparlar ve veri tabanındaki kart, müşteri ve işlem bilgilerini güncellerler.

Sorgulama ve güncelleme ekranları, veri tabanındaki kart, müşteri ve işlem bilgilerinin sorgulanmasına ve güncellenmesine olanak sağlarlar.

Bu çalışmada, işlem sunucusu üzerinde birtakım performans ölçümleri yapılmıştır. Bu ölçümlerin yapılabilmesi için istemci yönetim programı ve istemci program tasarlanmıştır. İstemci yönetim programı istemci programların başlatılmasından sorumludur. İstemci program ise işlem örnekleri veri tabanından okuduğu kayıtları kullanarak, işlem sunucusuna işlem istekleri gönderir. İşlem sunucusu üzerindeki yük, istemci program sayısına bağlıdır. Zaman içerisinde çalışan istemci program

sayısı değiştirilerek işlem sunucusunun üzerindeki yük arttırılabilir ya da azaltılabilir.

Yapılan testlerde, işlem sunucusunun veri tabanına bağlantı kurma yöntemleri değiştirilerek farklı yük koşulları altında ölçümler yapılmış ve bu ölçümler kullanılarak çeşitli grafikler elde edilmiştir. İşlem sunucusunda veri tabanı bağlantısı kurmak için 3 farklı yöntem uygulanmıştır. 1. yöntemde işlem sunucusu, açılırken veri tabanına belirli bir sayıda bağlantı kurar ve kapatılana kadar bağlantı sayısı sabit kalır. 2. yöntemde, işlem sunucusu açılırken veri tabanına başlangıç sayısı kadar bağlantı kurar ve anlık olarak kullanılan veri tabanı bağlantı sayısına göre dinamik olarak ayarlamalar yapar. 3. yöntemde veri tabanı bağlantısı her işlem isteği için kurulur ve işlem sonlandıktan sonra da kapatılır.

Her 3 yöntem için farklı yük koşulları altında yanıt süresinin, işlem sayısının, açılan ve kullanılan veri tabanı bağlantı sayısının zamana göre değişim grafikleri çıkarılmıştır. Oluşturulan grafikler 3 yöntem için karşılaştırılarak, yöntemlerin üstün ve zayıf özellikleri açıklanmıştır. Yöntemler arasında yapılan karşılaştırmalardan şu sonuç çıkmaktadır: Kullanılacak olan yöntem, uygulama ve veri tabanı üzerinde yapılan bir takım ölçümlerin değerlendirilmesi sonucunda seçilmelidir.

İşlem sunucusu uygulaması şu yöntemlerle daha fazla geliştirilebilir: İşlem sunucusu veri tabanı bağlantı sayılarının dinamik olarak belirlendiği yöntemde, mevcut yapıda anlık kararlar verilmektedir. Bunun yerine işlem sunucusu yoğunluğu ile ilgili tutulabilecek istatistiksel bilgilere dayanarak bağlantı sayıları ayarlanabilir. Veri tabanı bağlantılarının dinamik olarak ayarlandığı ya da sabit tutulduğu yöntemlerde, bağlantı temin edilirken ilk boş bulunan bağlantı kullanılmaktadır. Bu durumda bazı bağlantılar yoğun bir şekilde kullanılırken, bazı bağlantılar ise boş kalabilmektedir. Bu durumu önleyecek bir bağlantı temin algoritması geliştirilebilir. Mevcut yapıda tek bir bilgisayar üzerinde bir tane işlemci sunucusu çalışmaktadır. Bu durumda bilgisayarda çıkabilecek problemler durumunda işlem sunucusunun kullanılabilirliğinde azalma görülecektir. Bunu önlemek için işlem sunucusu birden fazla bilgisayarda, dağıtık yapıda çalışır hale getirilebilir. Bu durumda sunucunun ölçeklenebilirliği de artacaktır.

KAYNAKLAR

[1] **Cantu, M.,** 2001. Mastering Delphi 6, SYBEX Inc., Alameda CA.

[2] **Harmon, E.,** 2000. Delphi COM Programming, MTP, Indianapolis.

[3] **Introduction to Indy,** (2002), *Introduction to Indy,*
<http://www.hower.org/Kudzu/Articles/IntroToIndy>

[4] **ODBC Product Information,** (1997), *ODBC Product Information,*
<http://www.microsoft.com/data/odbc/prodinfo.htm>

[5] **OLE DB Product Information,** (1997), *OLE DB Product Information,*
<http://www.microsoft.com/data/oledb/prodinfo.htm>

[6] **ADO Product Information,** (1997), *ADO Product Information,*
<http://www.microsoft.com/data/ado/prodinfo.htm>

[7] **Introduction to SSL,** (1998), *Introduction to SSL,*
<http://developer.netscape.com/docs/manuals/security/sslin/contents.htm>

[8] **Wagner, D. And Schneier, B.,** 1997. Analysis of the SSL 3.0 protocol, University of California, Berkeley

ÖZGEÇMİŞ

Özgür Tabak 1976 yılı Zonguldak doğumludur. Ortaokul ve liseyi Zonguldak Atatürk Anadolu Lisesinde okudu. 1994 yılında buradan mezun oldu ve İstanbul Teknik Üniversitesi Elektrik Elektronik Fakültesi Kontrol ve Bilgisayar Mühendisliğinde lisans öğrenimine başladı. Lisans diplomasını 1998 yılında aldı. Aynı yıl İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol ve Bilgisayar Mühendisliği bölümünde yüksek lisans öğrenimine başladı. Şu an Koçbank Bilgi İşlem Merkezinde Proje Mühendisi olarak çalışmaktadır.