**PROGRAMLAMA LABORATUVARI 2**

**3. PROJE**

Hazar KOÇ – Zeynep Aslı ERHAN

*Bilgisayar Mühendisliği Bölümü*

*Kocaeli Üniversitesi*

[hazark41@gmail.com](mailto:hazark41@gmail.com) – [zerhanas@gmail.com](mailto:zerhanas@gmail.com)

**ÖZET**

Bu doküman Programlama Laboratuvarı 2 dersi 3. Projesi için çözümümüzü açıklamaya yönelik oluşturulmuştur. Dokümanda projenin tanımı, çözüme yönelik yapılan araştırmalar, kullanılan yöntemler, proje hazırlanırken kullanılan geliştirme ortamı ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projemizi hazırlarken kullandığımız kaynaklar ve proje derlenirken dikkat edilmesi gereken hususlar bulunmaktadır.

# I. PROJE TANIMI

Projede bizden istenen:

* Bir bankanın yönetim sistemi için bir veritabanı tasarlamak ve bu veritabanının üzerinde gerekli işlemleri gerçekleyen bir uyguluma gerçekleştirmek

# II. ARAŞTIRMALAR VE YÖNTEM

Banka içerisinde müşteri, temsilci ve banka müdürü olmak üzere 3 adet rol bulunmaktadır. Müşteriler ve çalışanlar için gerekli tanımlayıcı bilgiler (Ad, Soyad, Telefon, TC No, Adres, E-posta) veri tabanında saklanmalıdır. Bir müşterinin birden fazla hesabı bulunabilir. Hesaplar sistem içerisinde kayıtlı bulunan herhangi bir para birimi cinsinden açılabilir (TL varsayılan olarak gelmelidir). Hesaplar arası para transferinde gerekli durumlarda kur dönüşümü otomatik olarak yapılmaktadır. Tüm bu eylemlerin tasarlanan bir arayüz üzerinden görsel bir şekilde gösterilmiştir.

## Form Dizaynları

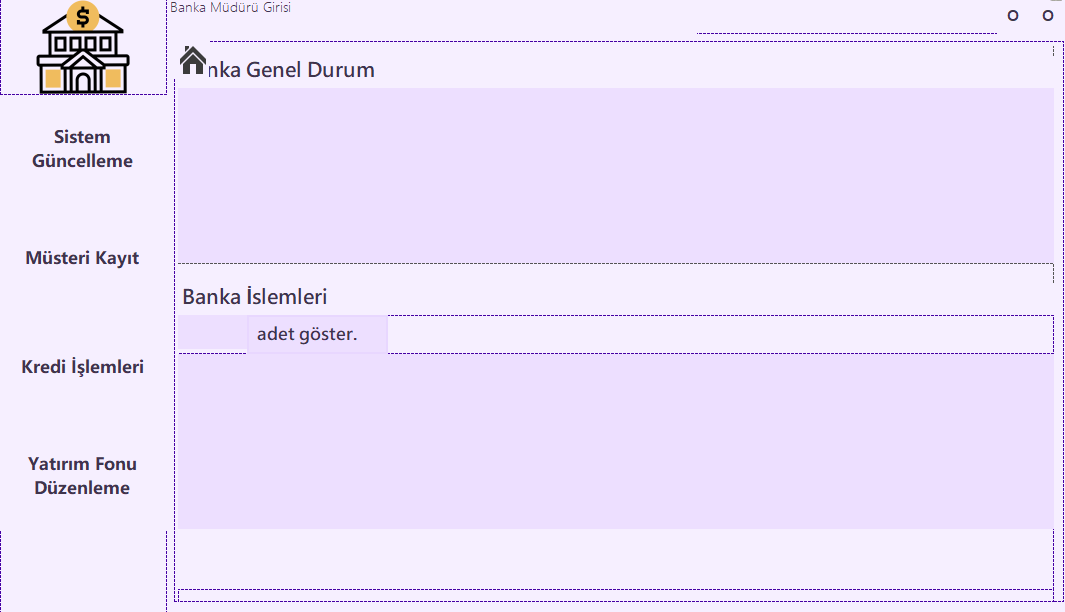
1 - **Login**



Müşteri için Giriş



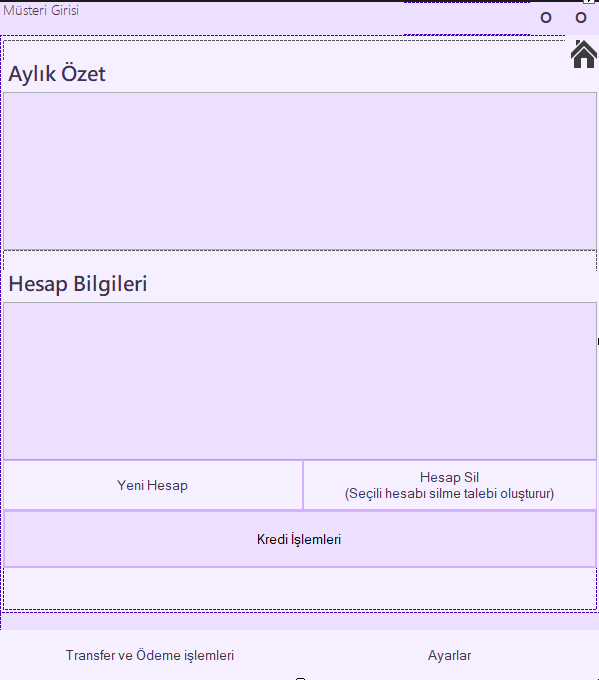
2 - **Banka Müdürü**



3 - **Müşteri Temsilcisi**

# 

4 - **Müşteri**



Veritabanı tasarım aşamasında ise:

Tablolarımızı normalizasyon formlarına göre hazırlayarak verilerin karşılaşabileceği anomalileri engelledik. Normalizasyon formlarına detaylıca değinmeden önce başlıca dikkate aldığımız anomalilerden bahsedelim:

Insertion Anomaly : Her müşterinin sahip olabileceği bir tane müşteri temsilcisi vardır. Bu bilginin saklanması için “cus\_list” tablosunda ilgili müşteri temsilcisinin bilgisi tutulmalıdır. Eğer ilgili müşteri temsilcisinin sadece bu tabloda detaylı bilgilerini tutarsak problem yaratır mı? İlk akla gelen sorun bankamıza yeni bir müşteri temsilcisi geldiği zaman hiç müşterisi olmayabilir ve tablodaki veriler “null” atanamıyorsa insertion anomaly’siyle karşılaşabileceğimiz oldu. Bu anomaliden korunmak için müşteri temsilci bilgileri “workers” tablosunda tuttuk. Müşteri temsilcisinin “workers” tablosundaki id’si, “cus\_list” tablosunda foreign key olarak kullandık. Artık her müşterinin bir tane müşteri temsilcisi, her müşteri temsilcisinin istediği sayıda müşterisi olmuş oldu.

Modification Anomaly : Müşterilerin birden daha fazla sayıda hesabı olabilir. Bu hesapların detaylı bilgileri ve kime ait olduğu “accounts” tablosunda tutulmaktadır. Hesabın kime ait olduğu verisi, hesap sahibinin ismi yapılırsa problem yaratır mı? Müşteriler kendi isimlerini ve müşteri temsilcileri ise ilgilendikleri müşterilerin isimlerini güncelleyebilirler. Bu gibi bir durumda o müşterinin sahip olduğu hesaptaki, hesap sahibinin isminin de güncellenmesi gerekir. Bu durumda o müşterinin birden fazla hesabı varsa ve herhangi bir hesabında güncelleme hatası olursa “modification anomaly” durumuyla karşılaşırız. Bu anomaliden kurtulmak için “cus\_list” tablosunun -primary key olan- “ id ” bilgisini “accounts” tablosunda foreign key olarak kullandık. Artık her hesap bilgisinde müşterilerin primary key özelliği taşıyan “id” bilgisi, isim değişikliği oldupunda hesapları etkilememektedir.

Deletion Anomaly : Müşterilerin sahip olabileceği yatırım fonu hesaplarında; seçtikleri dolar, euro, altın gibi çeşitli para birimlerini veya çeşitli yatırımları bulundurabilirler. Eğer yatırım fonu türlerinin detaylı bilgisini sadece müşterilerin açtığı yatırım hesabında tutarsak problem yaratır mı? Her yatırım fonu türünde hesabımız olursa, her yatırım fonu türü için detaylı bilgimiz olur fakat örneğin, dolar hesabı olan müşterilerin hepsi hesaplarını kapatırsa artık bankamızda yatırım fonu türü olarak dolar olmadığı için dolar kuru gibi detaylı bilgiler de olmaz ve deletion anomaly durumu ile karşılaşırız. Bu anomaliden kurtulmak için “yatırım\_fon” tablosunda her yatırım fonu türünün detaylı bilgilerini tuttuk. Primary key olan “para\_id” değişkenini accounts tablosunda foreign key olarak kullandık. Artık bankamızda kayıtlı veya yeni kaydedilecek herhangi bir yatırım fonu türünün bilgilerini koruyoruz.

Bu gibi anomaliler ve daha fazla hatalardan korunmak için normalizasyona ihtiyaç duyarız. Sırasıyla :

1. Normalizasyon Formu :

* Her tabloda Primary key bulunmalıdır.
* Bir tabloda aynı tür veriden birden fazla veri bulunmamalıdır.
* Her kolon sadece tek tip bilgi içermelidir.
* Tekrarlanan kolonlar olmamalıdır.

xxx

1. Normalizasyon Formu :

* 1. Normalizasyon formuna uymalı.
* Partial Dependery olmamalı.

Müşteri ile hesap tablosundaki veriler, müşteri tablosunda bir arada tutulsaydı; bir müşterinin bilgileri sahip olduğu hesap sayısı kadar tekrarlanırdı veya açılan hesap türleri de o hesap türüne sahip müşteri sayısı kadar tekrarlanırdı.

Yeni tablo için candiate key {müşteri\_id, hesap\_id} olur dolayısıyla primary key, {müşteri\_id, hesap\_id} olarak belirlenir.  
 Candiate key’lerin alt kümeleri non-prime özelliği tek başına belirleyebiliyorsa partial dependery durumu oluşur. Dolayısıyla {müşteri\_id} ile {hesap\_id} tek başına bir özelliği belirtebiliyor mu kontrol etmeliyiz.

Müşteri\_id -> müşteri\_tc belirleyebiliyor. Partial dependery durumundan korunmak için hesap bilgilerinin ayrı bir tabloda tutulması gerekir.

1. Normalizasyon Formu :

* 2. Normalizasyon formuna uymalı.
* Transitive Dependery olmamalı.

Hesaplar ve hesap türleri tablosundaki veriler aynı tabloda tutulsaydı yeni bir hesap türünün oluşturulması için müşterilerin yeni bir hesap oluştuması gerektiğinden bu işlem yapılamazdı. Aynı durum banka ile çalışanlar tablosu aynı tabloda tutulsaydı da olurdu. Çünkü bankamızda yeni bir meslek grubu (sekreter, güvenlik görevlisi vs) eklemek için bankada o işe sahip çalışan olması gerektiğinden işlem yapılamazdı.

# III. GELİŞTİRME ORTAMI

Projemizi Windows 10 sistemde, veritabanı için PostgreSQL; arayüz tasarımı için ise Visual Studio üzerinde C# kullanarak geliştirdik.

# IV. KOD BİLGİSİ

SÖZDE KOD

**Bağlantı oluşturup açma**

conn.Open() ile server’a bağlantı açıldı

sql stringine gerekli komut işlemi yazıldı

cmd değişkeni, yazılan sql stringi ve conn bağlantısı kullanılarak yeni bir sql komut oluşturuldu

cmd komutu sayesinde sql komutunda eklenmesi gereken değerler eklendi

Result cmd sayesinde ulaşılan verinin bilgisini tuttu

conn.Close() ile server bağlantısı kapatıldı

**Komut Oluşturma**

sql, stringij oluşturulur

cmd adında bir NpgsqlCommand oluşturulur

dt adında bir DataTable oluşturulur

//örnek

sql=@”select \* from müşteriler()”;

cmd= new NpgsqlCommand(sql,conn);

**DataGridView Veri Aktarımı**

dt adında bir DataTable oluşturulur

yeni DataTable() objesi dt’ye atanır(eşitlenir)

dt.Load() ile cmd.ExecuteReader() çağırılir ve okunan data yüklenir

conn.Close() ile server bağlantısı kapatıldı //DataGridView ismi dgvData

dgvData.DataSourse null’a eşitlenir yani sıfırlanır //reset

Boş dgvData.DataSource dt verilerini alır

**Select()** **fonksiyonu**

private void İlgiliMüşteriler\_Select()

{

try

{

conn.Open();

sql = @"SELECT \* FROM musteriler M

JOIN calisanlar C on C.id=M.temsilci\_id

WHERE C.tc=" + MusTemsilcisi\_tc ;

cmd = new NpgsqlCommand(sql, conn);

dt = new DataTable();

dt.Load(cmd.ExecuteReader());

conn.Close();

Dgv\_İlgilenilenMüşteriListesi.DataSource = null;//reset

Dgv\_İlgilenilenMüşteriListesi.DataSource = dt;

}

catch (Exception ex)

{

conn.Close();

MessageBox.Show("ERROR : "+ex.Message);

}

}

**Insert**

try

{conn.Open() ile servera bağlantı açıldı

Sql’in içine insert\_musteriler() fonksiyonu ile \_uye\_id, \_hesap\_tur\_id, \_yatirim\_fon\_id, \_bakiye ve \_hesap\_isim dataları girilir

cmd adında yeni NpgsqlCommand() objesi oluşturulur (İçine sql ve conn değerleri atanmış biçimde)

cmd.Parameters.AddWithValue() fonksiyonu ile verilen "\_yatirim\_fon\_id" değerine Yatırım Fonu Türü girilir

cmd.Parameters.AddWithValue() fonksiyonu ile verilen “\_yatirim\_fon\_id" değerine Hesap Türü değeri girilir

cmd.Parameters.AddWithValue() fonksiyonu ile verilen "\_hesap\_isim" değerine (yerine) TxtBox\_Hesapİsmi.Text ‘ten alınan değer girilir

cmd.Parameters.AddWithValue() fonksiyonu ile verilen “\_uye\_id"e Mus\_Tc değeri girilir (müşteri T.C.)

cmd.Parameters.AddWithValue() fonksiyonu ile verilen "\_bakiye" değerine 0 olarak atanır

result değeri içine girilen cmd.ExecuteScalar() integer olarak atanır

conn.Close() ile server bağlantısı kapatıldı

if ( eğer result 1 ise (exist ise)

"Hesap Oluşturuldu." yazısı MessageBox ile gösterilir

else:

"Hesap Oluşturulamadı." Yazısı MessageBox ile gösterilir

}

}

catch (Exception ex)/ YAKALA(hata ismi)

{

conn.Close() ile server bağlantısı kapatıldı

"Hesap Oluşturma Başarısız. Error: " + hata ismi yazısı MessageBox ile gösterilir

}

# KAYNAKLAR

[1] <https://www.youtube.com/watch?v=JFxY2qajjwA>

[2] <https://www.youtube.com/watch?v=URskGHtvFlE>

[3] <https://www.youtube.com/playlist?list=PLh9ECzBB8tJOS7WQKdeUaAa5fmPLYAouD>

[4] <https://www.youtube.com/watch?v=U_v1dSglNjE>