Banka Sistemi

Kocaeli Üniversitesi Bilgisayar Mühendislik Bölümü Programlama Laboratuvarı II Proje 3

200201137 Marah Alasi — 190201140 Mohamed Hosam Mohamed Gomaa Helwa

15.05.2022

Özet

Bu projede bir bankanın yönetim sistemi için bir veritabanı tasarlanması ve bu veritabanının üzerinde gerekli işlemleri gerçekleyen bir uygulama geliştirilmesi amaçlanmaktadır.

---

1 Giriş

Tasarlanan veritabanı, bankanın müşterilere, çalışanlara, hesaplara ve işlemlere ilişkin bilgileri organize bir şekilde işlemesine yardımcı olacaktır. Bu şekilde bankanın ve müşterilerin ihtiyacı olan bilgilere daha kolay ulaşabilmesi sağlanmış olacaktır. Ayrıca, veritabanı kullanarak müşteri istekleri ve bankanın ihtiyaçları doğrultusunda raporlar hazırlanabilecektir.

2 Önbilgi

2.1 Bankadaki roller

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 yapılmalıdır. Rollerin gerçekleştirdiği eylemler aşağıda belirtilmiştir. Tüm bu eylemlerin tasarlanan bir arayüz üzerinden görsel bir şekilde gösterilecektir.

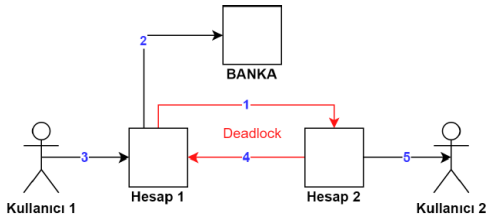
1. Müşteriler:
    - Hesaplarından para çekebilirler ve yatırabilirler.
    - Yeni hesap açma ve var olan bir hesabı silme talebinde bulunabilirler.
    - Birbirleri arasında para transferi yapabilirler.
    - Bilgilerini güncelleyebilirler. (Adres, Telefon vs.)
    - Bankaya para transferi yapabilirler. (Kredi borcu ödeme)
    - Bankadan kredi talep edebilirler.
    - Aylık özetlerini görüntüleyebilirler. (Geçerli ay içerisinde yaptığı para gönderme, çekme, kredi borcu ödeme gibi işlemlerin özeti)
2. Müşteri Temsilcisi:
    - Her müşterinin bir temsilcisi vardır.
    - Müşteri ekleme, silme ve düzenleme yapabilir (silme ve düzenleme işlemleri sadece kendi müşterileri için geçerlidir).
    - Müşteri bilgilerini güncelleyebilirler. (Adres, Telefon vs.)
    - Bilgilerini güncelleyebilirler. (Adres, Telefon vs.)
    - İlgilendikleri müşterilerin genel durumlarını (gelir, gider ve toplam bakiye) görüntüleyebilmektedir.
    - Müşterilerden gelen hesap açma, silme ve kredi taleplerini görüntüleme ve onaylama sorumluluğu temsilcilere aittir.
    - İlgilendikleri müşterilerin işlemlerini (para çekme, yatırma ve transfer) görüntüleyebilmektedir.
3. Banka Müdürü:
    - Bankanın genel durumunu (gelir, gider, kar ve toplam bakiye) görüntüleyebilmektedir.
    - Yeni para birimi (Dolar, Euro, Sterling vs.) ekleyebilir ve kur değerlerini güncelleyebilir.
    - Çalışanların maaş ücretlerini belirleyebilecektir.
    - Kredi ve gecikme faiz oranını belirler.
    - Müşteri ekleyebilir.
    - Sistemi bir ay ilerletebilir.
    - Bankada gerçekleşen tüm işlemleri (para çekme, yatırma ve transfer) görüntüleyebilmektedir.
    - Listelenen işlemlerin aynı anda başlatılması durumunda deadlock oluşup, oluşmadığının analizinin yapabilmektedir

2.2 Deadlock:

Para gönderimi sırasında hedefin işlem yapması engellenmektedir. Bu nedenle para almakta olan bir hesap para gönderimi yapamamaktadır. Deadlock analizi için tüm işlemlemlerin aynı anda çalışmaya başladığı ve paralel şekilde çalıştığı kabul edilecektir.

Şekilde görüldüğü gibi kullanıcı 1 kullanıcı 2'ye paray transferi yapıyor aynı şekilde kullanıcı 2 de kullanıcı 1'e para transferi yapmaktadır. Bu durumda 1.işlem 4.işlemi bitmesine beklerken aynı şekilde 4.işlem 1.işlemin bitmesine beklimektedir, iki işlem birbirllerinin bitmesine beklediği için ikisi sonsuza kdar bekler ve program hata ile

sonuçlandırılır. Bu durumu banka hareketlerinde banka müdür tarafından aşağıdaki gibi görüntülenecektir.



Deadlock sayısı : 1

İşlemler : (1-4)

2.3 Kredi

Müşteriler Bankadan kredi isteyebilirler. Kredi norçlarını ve oluşan faizleri görüntülemek için bir tablo oluşturması amaclanmıştır. Talep ettikleri kredi borç miktarı gerekli faiz hesaplamaları ile beraber aylara bölünerek gösterilecektir. Müşterinin talep ettiği borç ilk önce ay sayısına bölünür ve buna anapara denir sonra faiz yüzdesi ile çarpılır ve buna faiz denir. Bu 2 değer (artı gecikme faizi ama bu değer önceki aylarda ödenen miktara bağlı olduğu için 1. Ayda gecikme miktarı 0'dır) şu anki ay için en az ödemesi gereken miktardır ve buna ödeme payment denir. Left Kalan miktar ise toplam borçtan ödemesi gereken kalan miktardır. Bu değer şöyle hesaplanır: alınan tam borç (1. Ayda alınan tam borç 2. Ayden itibaren önceki ayın kalan miktarı ile hesaplanır) eksi o ayda ödenen para artı faiz ve gecikme faizi kalan miktara eşittir. Anapara 2. Aydan itibaren Kalan Miktar'a göre değişmeye başlar: yeni anapara değeri önceki ayın kalan para değeri bölü kalan ay sayısıdır.

Ay	Anapara	Faiz	Gecikme Faizi	En Az Ödeme	Ödenen	Kalan Miktar
1	5000	250	0	5250	5250	55000
2	5,000.00	250	0	5,250.00	5,250.00	50,000.00
3	5000	250	0	5250	6000	44,250.00
4	4916.666667	245.8333333	0	5162.5	3000	41,495.83
5	5186.979167	259.3489583	64.875	5511.203125	5511.203125	36,308.85
...	...	...	...	...	...	...

3 Yöntem

3.1 Veritabanı

Bir veritabanı tasarımının ilk aşamasında sistemin ihtiyaçlarının belirlenmesi ve depolanacak bilgi türlerinin tanımlanması için Varlık-İlişki (ER) diyagramı oluşturulmalıdır. ER diyagramı sistem içerisinde var olabilecek varlıkların ve aralarındaki ilişkilerin görsel olarak ifade edilmesi için kullanılır. Geliştirme sırasında, ER diyagramı gereksinimlerin daha açık ve özlü bir şekilde haritalanmasına yardımcı olmaktadır. Veritabanındaki tablolar en az 3NF normalizasyon'a uyacak şekilde tasarlanması gerekmektedir.Veritaban normalizasyon adımları section da yer almaktadır.

Oluşturulan veritabanı müşteri, müşteri temsilcisi, hesaplar, işlemler, para birimi ana tablolarından oluşmaktadır. Müşteri tablosunda müşterilerin genel bilgileri ve müşteri numaraları, müşteri temsilci tablosunda temsilcilerin temsilci numaraları, hesaplar tablosunda hesapların hesap numarası, para birimi ve bakiyesi, işlemler tablosunda işlem numarası, tarih, hedef ve kaynak hesapları, tutar, işlem türü ve hedef ve kaynağın işlemden sonraki bakiyeleri, para birimi tablosunda para birimin kodu ve TL'ye göre değişim oranları saklanmaktadır.

Müşteri hangi temsilciye ait olduğunu göstermek için bir müşteri-temsilci ilişki tablosu oluşmaktadır. Tabloda her satırda müşterinin müşteri no'su ve temsilcinin temsilci no'su

saklanmaktadır. Her iki değer de bir müşteri ve temsilci tablolardan alınan FOREIGN KEY'dir. Aynı şekilde müşteri-hesap ilişki tablosu da müşteri no ve hesap no sutünlerinden oluşup müşteri ve hesaplar tablolarından alınan FORGEIN KEY'lerdir.

Temsilci tarafından müşteri eklenme durumunda müşterinin aktive edilmesi için banka müdüründan onay gerekemektedir. müşteri veya hesap silindiğinde işlemler tablosunda iligişi işlemlerin silinmesi istenmemektedir. Bu durumlarda müşteri veya hesap silindiğinde veritabanından tamamen silinmesi istenmemektedir. Bunu önlemek için müşteri durumu ve hesap durumu tabloları oluşturulmustur. müşteri durumu tablosunda müşteri no'su ve durumu vardır, hesap tablosunda da aynı şekilde hesap no ve durum sutünleri vardır. Yeni müşteri eklendikten sonra ve banka müdüründen onay alınmadan önce müşteri bu tabloda 'NEW' yani yeni olarak kaydı alınmaktadır. Onay alınması durumunda müşterinin durumu değişğir ve 'ACTIVE' aktif olarak güncellenmektedir. Aynı durumlar hesap durumu tablosu için de geçerlidir.

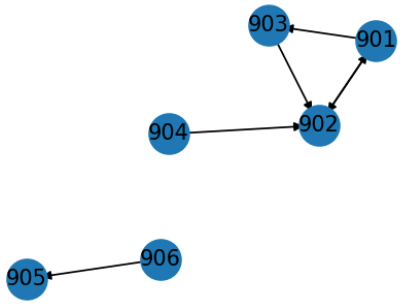
Hesap durumu tablosu hesap açma ve silme işlemleri için de gereklidir. Bir müşteri yeni hesap açma talep ettiğinde tabloda oluşturulan yeni hesabın hesap no'su ile beraber durumu 'R-OPEN' (açma talebi) olarak kaydedilmektedir. Silme işlemi için ise 'R-DELETE' olarak kaydedilmektedir.

3.2 Deadlock Analizi

Deadlock oluşan işlemleri bulmak için graf kullanılması tercih edildi. Bunun nedeni, aralarında deadlock oluşan sadece 2 değil 2'den fazla hesap arasında olabilir. Mesela aşağıdaki tabloda bulunan işlemler arasında deadlock'leri bulmaya çalışılırsa, 1. Deadlock 9001 ve 9001 arasında olduğunu buluruz, fakat 2. Bir deadlock vardır, o da 9001-9002-9003 arasındadır. 9001 9002'ya para gönderiyor, 9002 hem 9001'e hem de 9003'e gönderiyor ve sonunda 9003 9001' göndermektedir. Son işlem grafta bir döngü oluşmasına neden olumaktadır. Dolayısıyla, graf'taki tüm dögüleri deadlock olarak sayılabilir ve kod bunu göz önünde bulunarak yazılmalıdır.

işlemler tablosu

Kaynak	Hedef
9001	9002
9002	9001
9001	9003
9003	9002
9004	9002
9006	9005



4 Kaynakça

- https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/
- https://www.guru99.com/database-normalization.html

5 Normalizasyon adımları

1.adım

firstN	lastN	phone	Address	Account	account_status	customer_status	clerk_id
Robert	Williams	850-336-8915	B City	9003	ACTIVE	ACTIVE	1
Mary	Brown	936-891-0789	C City	9001	ACTIVE	ACTIVE	2
Bob	Jones	128-353-4436	F City	9004	DELETED	DELETED	4
Bob	Jones	87979789	F City	9005	R-OPEN	NEW	2

2.adım : müşterileri ayırt etmek için cus_id eklenmiştir

cus_id	firstN	lastN	phone	Address	Account	clerk_id
301	Robert	Williams	850-336-8915	B City	9003	1
302	Mary	Brown	936-891-0789	C City	9001	2
303	Bob	Jones	128-353-4436	F City	9004	4
304	Bob	Jones	87979789	F City	9005	2

3.adım : müşterilerin temsilcileri ve hesapları müşterilerin cus_id 'si ile birlikte ayrı bir tabloya konulmaktadır

cus_id	clerk_id
301	1
302	2
303	4
304	2

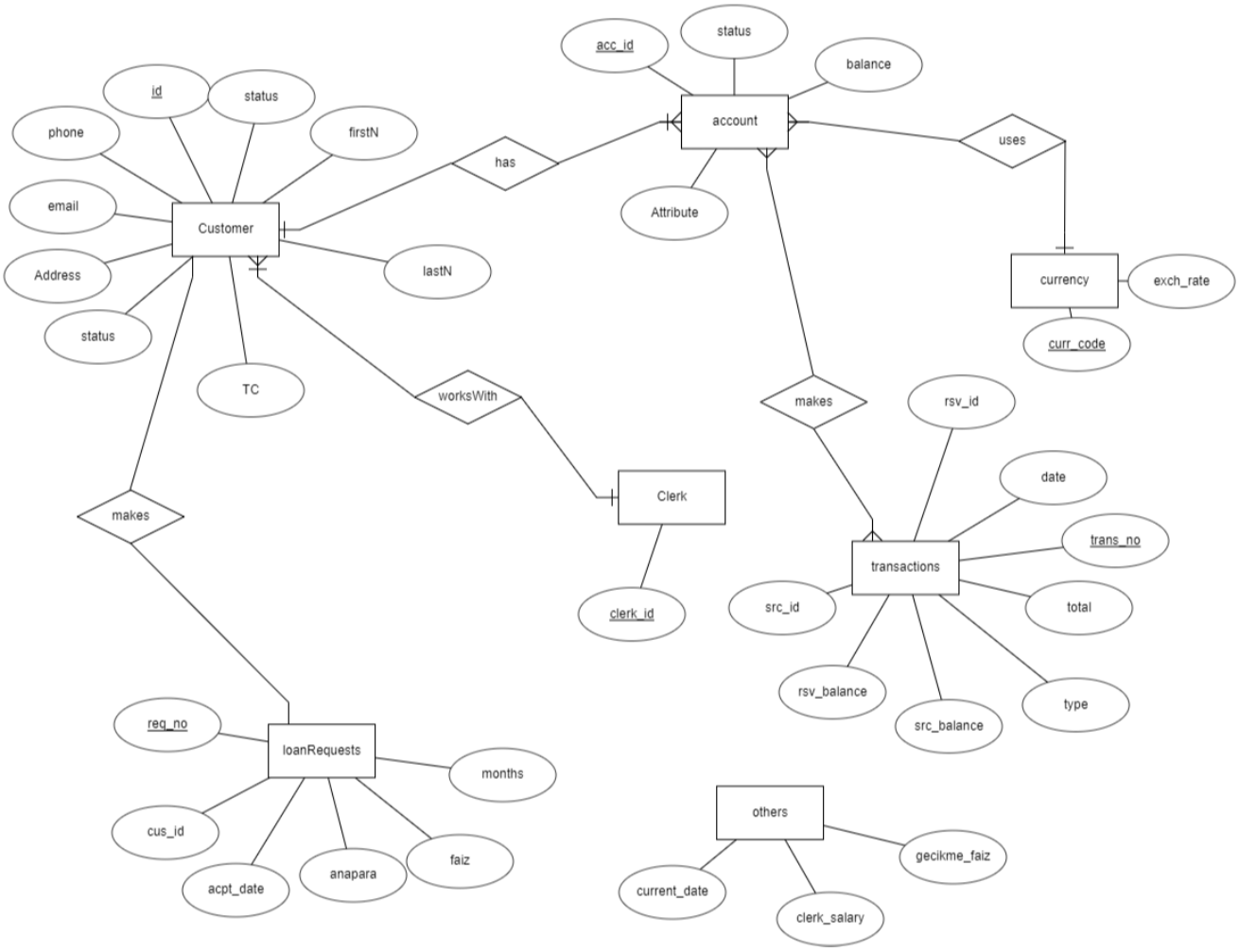
cus_id	Account
301	9003
302	9001
303	9004
304	9005

4.adım : müşteri ve hesapların durumları ayrı tablolara konulmuştur

cus_id	customer_status
301	ACTIVE
302	ACTIVE
303	DELETED
304	NEW

Account	account_status
9003	ACTIVE
9001	ACTIVE
9004	DELETED
9005	R-OPEN

ER Diyagramı



Algorithm 1

```
function findDeadLocks(limit)
    arr ← SQL Query "get top (limit) transactions"
    G ← create graph from arr
    cycles ← find cycles in G
    deadlocks : array
    for cycle in cycles do
        deadlock : array
        for src in cycle do
            transNo ← SQL Query "get transactions where srcid = src and rsvid = rsv"
            if transNo then
                deadlock_i ← transNo
            end if
        end for
        deadlocks_j ← deadlock
    end for
    return deadlocks
end function
```

Algorithm 2

function GETLOANINFO(cusID)
    $loans$: array
    $loanRow \leftarrow$ loanRequest from loanRequests where cusID = cusID
    $gecikmeFaiz \leftarrow$ get gecikme faiz from others
    $paidQuery \leftarrow$ get sum of total from transactions whhere cusID = cusID and transType='loan payment' and transDate between acceptedDate and currentDate
    if loanRow **then**
        $nextMonth \leftarrow$ currentDate + 1 Month
        $left \leftarrow loanRow.anapara$
        for i in range loanDuration(months) **do**
            arr : array
            $anapara \leftarrow left/(laonRow - i)$
            $faiz \leftarrow (anapara \times loanRow.faiz)/100$
            $k \leftarrow payment - paid$
            if $k \leq 0$ **then**
                $gecikme \leftarrow 0$
            else
                $gecikme \leftarrow k \times (gecikmeFaiz/100)$
            end if
            $payment \leftarrow anapara + faiz + gecikme$
            $arr \leftarrow$append nextMont, anapara, faiz, gecikme, payment, paid, left
            $loans \leftarrow$ append arr
        end for
    else
        return None
    end if
    return loans
end function