PROLAB2-3

Ogün Bingöl-Bünyamin Ekşici
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
ogunbingol52@gmail.com
bunyamineksici@gmail.com

***Abstract*—Bu projede bir bankanın yönetim sistemi için bir veritabanı tasarlamanız ve bu veritabanının üzerinde gerekli işlemleri gerçekleyen bir uygulama geliştirmeniz amaçlanmaktadır.**

***Index Terms*—banka yönetim sistemi,java gui,mysql , database processes**

I. 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. 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. Sizlerden problemlerin çözümü için bir veritabanı tasarımı yapmanız istenmektedir. Tablo sayısı tüm tablolar en az 3NF normalizasyon formuna uyacak şekilde size bırakılmıştır. Projede oluşturulacak veritabanının ER diyagramının oluşturulması ve arayüzde gösterilmesi gerekmektedir. Diyagram üzerinden gerçekleştirilen normalizasyon işlemlerinin adım adım gösterilmelidir. Veritabanı kesinlikle yazılım sistemlerinin ayrılmaz bir parçasıdır. ER Şemasını veritabanı mühendisliğinde tam olarak kullanmak, veritabanı oluşturma, yönetim ve bakımda kullanmak üzere yüksek kaliteli veritabanı tasarımı üretmenizi garanti eder. Bir ER modeli ayrıca iletişim için bir araç sağlar.

ERD, ER Şeması veya ER modeli olarak da bilinen Varlık İlişki Şeması, veritabanı tasarımında kullanım için bir tür yapısal diyagramdır. Bir ERD, iki önemli bilgiyi görselleştiren farklı semboller ve bağlayıcılar içerir: Sistem kapsamındaki ana varlıklar ve bu varlıklar arasındaki ilişkiler .

İşte bu yüzden buna ”Varlık” ”İlişki” diyagramı (ERD) deniyor!

ERD'deki varlıklar hakkında konuştuğumuzda, genellikle insanlar/roller (örn. Öğrenci), somut iş nesneleri (örn. Ürün), soyut iş nesneleri (örn. Log), vb. gibi iş nesnelerinden bahsediyoruz. bu varlıklar sistem içinde birbirleriyle ilişkilidir.

Tipik bir ER tasarımında, varlıkları, niteliklerini ve aralarındaki ilişkileri gösteren yuvarlak dikdörtgenler ve bağlayıcılar (uçlarının farklı stilleri olan) gibi semboller bulabilirsiniz.

ER Diyagramları ne zaman çizilir? Peki, ERD'leri ne zaman çiziyoruz? ER modelleri çoğunlukla kavram görselleştirme ve fiziksel veritabanı tasarımı açısından ilişkisel veritabanları tasarlamak için geliştirilmiş olsa da, ER diyagramlarının yardımcı olabileceği başka durumlar da vardır. İşte bazı tipik kullanım durumları.

1) Veritabanı tasarımı - Değişimin ölçeğine bağlı olarak, bir veritabanı yapısını doğrudan bir VTYS'de değiştirmek riskli olabilir. Bir üretim veritabanındaki verileri bozmamak için değişiklikleri dikkatli bir şekilde planlamak önemlidir. ERD yardımcı olan bir araçtır. Veritabanı tasarım fikirlerini görselleştirmek için ER diyagramları çizerek, hataları ve tasarım kusurlarını belirleme ve veritabanındaki değişiklikleri uygulamadan önce düzeltmeler yapma şansınız olur.
2) Veritabanı hata ayıklama - Özellikle veritabanı, ihtiyacınız olan bilgiyi elde etmek için karmaşık SQL yazmayı gerektiren birçok tablo içerdiğinde, veritabanı sorunlarını ayıklamak zor olabilir. Bir ERD ile bir veritabanı şemasını görselleştirerek, tüm veritabanı şemasının tam bir resmine sahip olursunuz. Varlıkları kolayca bulabilir, niteliklerini görüntüleyebilir ve başkalarıyla olan ilişkilerini tanımlayabilirsiniz. Tüm bunlar, mevcut bir veritabanını analiz etmenizi ve veritabanı sorunlarını daha kolay ortaya çıkarmanızı sağlar.
3) Veritabanı oluşturma ve yamalama - Bir ERD aracı olan Visual Paradigm, ER diyagramları aracılığıyla veritabanı oluşturma ve yama sürecini otomatikleştirebilen bir veritabanı oluşturma aracını destekler. Bu nedenle, bu ER Diyagramı aracıyla, ER tasarımınız artık yalnızca statik bir diyagram değil, fiziksel veritabanı yapısını gerçekten yansıtan bir aynadır.
4) Gereksinim toplamada yardım - Sistemin üst düzey iş nesnelerini gösteren kavramsal bir ERD çizerek bir bilgi sisteminin gereksinimlerini belirleyin. Böyle bir başlangıç modeli, ilişkisel bir veritabanının oluşturulmasına yardımcı olan veya süreç haritalarının ve veri akış modlarının oluşturulmasına yardımcı olan

bir fiziksel veritabanı modeline de evrilebilir.

Varlık
Bir ERD varlığı, bir kişi/rol (ör. Öğrenci), nesne (ör. Fatura), kavram (ör. Profil) veya olay (ör. İşlem) gibi bir sistem içindeki tanımlanabilir bir şey veya kavramdır (not: ERD'de " varlık" genellikle "tablo" yerine kullanılır, ancak bunlar aynıdır). Varlıkları belirlerken onları isim olarak düşünün. ER modellerinde, bir varlık, adı üstte ve nitelikleri varlık şeklinin gövdesinde listelenmiş olarak yuvarlatılmış bir dikdörtgen olarak gösterilir.

Varlık Özellikleri
Sütun olarak da bilinen öznitelik, onu tutan varlığın bir özelliği veya özelliğidir .

Bir özniteliğin, özelliği tanımlayan bir adı ve bir dize için varchar ve tamsayı için int gibi öznitelik türünü tanımlayan bir türü vardır. Fiziksel veritabanı geliştirme için bir ERD çizildiğinde, hedef RDBMS tarafından desteklenen türlerin kullanımını sağlamak önemlidir.

Birincil anahtar
PK olarak da bilinen birincil anahtar, bir veritabanı tablosundaki bir kaydı benzersiz olarak tanımlayan özel bir tür varlık niteliğidir . Başka bir deyişle, birincil anahtar özniteliği için aynı değeri paylaşan iki (veya daha fazla) kayıt olmamalıdır. Aşağıdaki ERD örneği, birincil anahtar özniteliği 'ID' olan bir 'Ürün' varlığını ve veritabanındaki tablo kayıtlarının önizlemesini gösterir. Üçüncü kayıt geçersiz çünkü 'PDT-0002' kimliğinin değeri zaten başka bir kayıt tarafından kullanılıyor.

Yabancı Anahtar
FK olarak da bilinen yabancı anahtar, bir tablodaki birincil anahtara yapılan başvurudur . Varlıklar arasındaki ilişkileri tanımlamak için kullanılır. Yabancı anahtarların benzersiz olması gerekmediğini unutmayın. Birden çok kayıt aynı değerleri paylaşabilir. Aşağıdaki ER Diyagramı örneği, aralarında başka bir varlığa atıfta bulunmak için bir yabancı anahtarın kullanıldığı bazı sütunlara sahip bir varlığı göstermektedir.

İlişki
İki varlık arasındaki ilişki, iki varlığın bir şekilde birbiriyle ilişkili olduğunu gösterir . Örneğin, bir öğrenci bir kursa kayıt olabilir. Bu nedenle Student varlığı Course ile ilişkilidir ve bir ilişki, bunlar arasında bağlantı kuran bir bağlayıcı olarak sunulur.

Kardinalite
Kardinalite, bir varlıktaki olası oluşum sayısını, diğer bir varlıktaki oluşum sayısıyla ilişkili olarak tanımlar . Örneğin, BİR takımın BİRÇOK oyuncusu vardır. Bir ERD'de mevcut olduğunda, Takım ve Oyuncu varlığı bire çok ilişkisiyle birbirine bağlıdır.

Bir ER diyagramında, kardinalite, bağlayıcının uçlarında bir kaz ayağı olarak temsil edilir. Üç yaygın kardinal ilişki bire bir, bire çoğa ve çoktan çoğadır.

Bire Bir kardinalite örneği
Bire bir ilişki, çoğunlukla bilgiyi kısaca sağlamak ve daha anlaşılır kılmak için bir varlığı ikiye bölmek için kullanılır. Aşağıdaki şekil bire bir ilişki örneğini göstermektedir.
Bire Çok kardinalite örneği
Bire-çok ilişki, X ve Y varlıkları arasındaki, X'in bir örneğinin birçok Y örneğine bağlı olabileceği, ancak bir Y örneğinin yalnızca bir X örneğine bağlı olduğu ilişkiyi ifade eder. Aşağıdaki şekil gösterilmektedir. bire çok ilişkisine bir örnek.
Çoktan Çoka kardinalite örneği
Çoktan çoğa ilişki, X'in birçok Y örneğine bağlı olabileceği iki varlık X ve Y arasındaki ilişkiyi ifade eder ve bunun tersi de geçerlidir. Aşağıdaki şekil çoktan çoğa ilişki örneğini göstermektedir. Fiziksel bir ERD'de çoktan çoğa ilişkinin bir çift bire çok ilişkiye bölündüğünü unutmayın. Bir sonraki bölümde fiziksel bir ERD'nin ne olduğunu bileceksiniz.

Bir ER modelinin üç düzeyinin tümü, niteliklere ve ilişkilere sahip varlıklar içerirken, oluşturuldukları amaçlar ve hedeflemeleri gereken hedef kitleler açısından farklılık gösterirler.

Üç veri modeline ilişkin genel bir anlayış, iş analistinin sistemde var olan iş nesnelerini modellemek için kavramsal ve mantıksal bir model kullanması, veritabanı tasarımcısı veya veritabanı mühendisinin ise fiziksel durumu sunan fiziksel modeli üretmek için kavramsal ve mantıksal ER modelini geliştirmesidir.
Kavramsal veri modeli
Kavramsal ERD , bir sistemde olması gereken iş nesnelerini ve bunlar arasındaki ilişkileri modeller . İlgili iş nesnelerini tanıyarak sistemin genel bir resmini sunmak için kavramsal bir model geliştirilmiştir. Hangi tabloların DEĞİL, hangi varlıkların var olduğunu tanımlar. Örneğin, mantıksal veya fiziksel bir veri modelinde 'çoktan çoğa' tablolar mevcut olabilir, ancak bunlar kavramsal veri modeli altında yalnızca kardinalitesi olmayan bir ilişki olarak gösterilir.

Mantıksal veri modeli
Mantıksal ERD, Kavramsal ERD'nin ayrıntılı bir sürümüdür . Her varlıktaki sütunları açıkça tanımlayarak ve operasyonel ve işlemsel varlıkları tanıtarak kavramsal bir modeli zenginleştirmek için mantıksal bir ER modeli geliştirilmiştir. Mantıksal bir veri modeli, veritabanının oluşturulacağı gerçek veritabanı sisteminden hala bağımsız olsa da, tasarımı etkiliyorsa bunu yine de göz önünde bulundurabilirsiniz.

Fiziksel veri modeli
Fiziksel ERD, ilişkisel bir veritabanının gerçek tasarım planını temsil eder . Fiziksel bir veri modeli, her bir sütuna tür, uzunluk, null yapılabilir vb. değerler atayarak mantıksal

veri modelini detaylandırır. Fiziksel bir ERD, verilerin belirli bir VTYS'de nasıl yapılandırılması ve ilişkilendirilmesi gerektiğini temsil ettiğinden, veritabanının oluşturulacağı gerçek veritabanı sistemi. Sütun türlerinin VTYS tarafından desteklendiğinden ve varlık ve sütunların adlandırılmasında ayrılmış sözcüklerin kullanılmadığından emin olun.

II. YÖNTEM

Problem Tanımı: 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 gösterilmesi gerekmektedir. Sistemdeki Roller: Müşteriler; o Hesaplarından para çekebilirler ve yatırabilirler. o Yeni hesap açma ve var olan bir hesabı silme talebinde bulunabilirler. Bakiyesi "0" olmayan bir hesap silinemez. o Birbirleri arasında para transferi yapabilirler. Farklı para birimlerine sahip hesaplar arası transferler sırasında gönderilen miktar hedef para birimine otomatik olarak çevrilmelidir. o Bilgilerini güncelleyebilirler. (Adres, Telefon vs.) o Bankaya para transferi yapabilirler. (Kredi borcu ödeme) o Bankadan kredi talep edebilirler. Kredi sadece TL cinsinden talep edilebilmektedir. Bankanın kredi talebini onaylaması durumunda istenilen vade oranınca (faiz ve anapara toplamı) bölünerek aylara borç olarak yansıtılır. Aylık özet görüntülemede kredi borcu ödemeleri için ödenen faiz ve anapara ayrı ayrı görüntülenmelidir. Müşterinin aylık borcunun tamamını ödememesi durumunda kalan borç ek faiz hesaplanarak bir sonraki aya devreder. Faiz ve gecikme faiz oranı banka müdürü tarafından belirlenir. Aylık borç ve kalan borç ayrı ayrı görüntülenmeli. (müşteri isterse tüm borcunu tek seferde ödeyebilir) Erken ödeme durumlarında gelecek aylar için faiz alınmayacaktır. o Aylık özetlerini görüntüleyebilirler. (Geçerli ay içerisinde yaptığı para gönderme, çekme, kredi borcu ödeme gibi işlemlerin özeti) Banka müdürü; o Bankanın genel durumunu (gelir, gider, kar ve toplam bakiye) görüntüleyebilmektedir. o Yeni para birimi (Dolar, Euro, Sterling vs.) ekleyebilir ve kur değerlerini güncelleyebilir. o Çalışanların maaş ücretlerini belirleyebilecektir. Tek bir çalışan türü vardır (müşteri temsilcisi). Hepsinin maaş miktarı aynıdır. o Kredi ve gecikme faiz oranını belirler. o Müşteri ekleyebilir. Sisteme yeni bir müşteri eklenmesi durumunda en az müşteriye sahip olan temsilciye atanır. o Sistemi bir ay ilerletebilir. İsterlerin test edilebilmesi için sizlerden uygulama tarihini bir ay ileriye öteleyebilmeniz istenmektedir. Bu ilerletme işlemi sonucunda -¿ maaşların ödenmesi, gelir-gider durumlarının güncellenmesi ve müşterilerin bir sonraki aya ait borçlarının kendilerine yansıtılması gerekmektedir. o Bankada gerçekleşen tüm işlemleri (para çekme, yatırma ve transfer) görüntüleyebilmektedir. İşlemleri listelerken "son X adet işlemi listele" şeklinde bir seçenek sunulmalıdır. Örnek olarak "son 5 işlemi listele" sorgusunun çıktısı Tablo I'de gösterilmiştir. o Listelenen işlemlerin aynı anda başlatılması durumunda deadlock oluşup, oluşmadığının analizinin yapabilmektedir. Deadlock analizi ayrı bölümde açıklanacaktır.

ALGORITMA :

Adım 1: Müşteri id bulan sorgu gönderilir
Adım 2: müşteri id'sinden hesap no bulan sorgu gönderilir
Adım 3: buradan hesap birim id bulan sorgu gönderilir
Adım 4: hesapturu tablosundan bakiyeyi bulan sorgu gönderilir
Adım 5: arayüzde çek butonuna basıldığında bakiye alınır
Adım 6: kullanıcının girdiği miktar alınır
Adım 7: kullanıcının girdiği miktar int değere dönüştürülür
Adım 8: bakiye miktar kontrolü yapılır
Adım 9: uygunsa bakiyeden miktar çıkarılır
Adım 10: bakiyeyi güncelleyen sorgu gönderilir

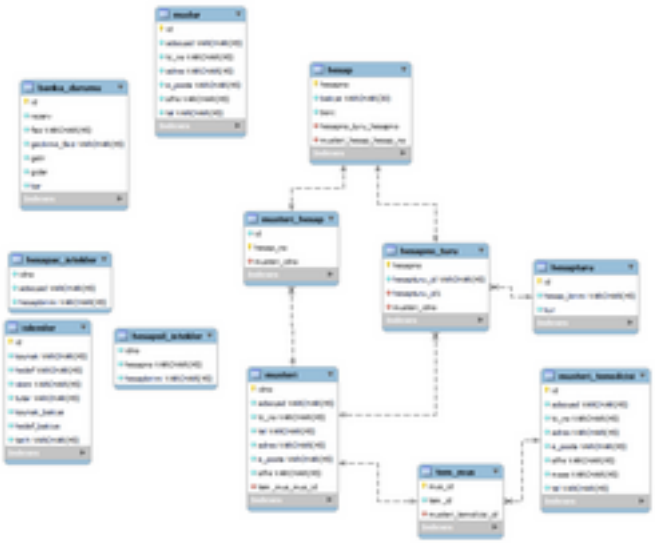

Fig. 1. Flame sensor uygulaması örneği

Müşteri temsilcisi; o Her müşterinin bir temsilcisi vardır. o Müşteri ekleme, silme ve düzenleme yapabilir (silme ve düzenleme işlemleri sadece kendi müşterileri için geçerlidir). o Müşteri bilgilerini güncelleyebilirler. (Adres, Telefon vs.) o İlgilendikleri müşterilerin genel durumlarını (gelir, gider ve toplam bakiye) görüntüleyebilmektedir. o Müşterilerden gelen hesap açma, silme ve kredi taleplerini görüntüleme ve onaylama sorumluluğu temsilcilere aittir. o İlgilendikleri müşterilerin işlemlerini (para çekme, yatırma ve transfer) görüntüleyebilmektedir. Deadlock Analizi: 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.

SONUÇ

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. 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.

REFERENCES

[1] https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/
[2] https://www.guru99.com/database-normalization/
[3] https://www3.ntu.edu.sg/home/ehchua/programming/java/
[4] https://www.javatpoint.com/java-swing