8. İHTİYACA ÖZEL VERİ TABANI TASARLAMAK

Birlikte Düşünelim

Yazılıma ihtiyaç duyan biri ve yazılımcı bir araya gelerek hazırlanacak sistemi birlikte tasarlayabilirler mi? Yoksa bu tasarlama süreci için farklı kişilere de ihtiyaç var mıdır?

Sistem analizi ve tasarımı süreçlerine hakim personel, bir yazılım firması için ne önemdedir?

Yazılım geliştirilerek çözülebilecek bir problem nasıl ele alınmalıdır?

Veri tabanı tasarımı sürecinde doğru soruları sormanın önemi nedir?

Başlamadan Önce

Dersin bu aşamasına kadar veri tabanı tasarımıyla ilgili çok şey öğrendiniz. Veri tabanlarına dair hiçbir bilgi ya da deneyimi olmayanların bile ilk bölümden itibaren veri tabanını tanıma, dokümante edebilme, kavramlara hakim olma, veri kategorizasyonu sağlama, tabloları ve ilişkileri oluşturma, özel durumlar için çözümler üretebilme gibi yeteneklere sahip olduğunu düşünüyorum. Şimdi her şeyi birleştirme zamanı. Bu dersin amacı, dersi alanların ihtiyaca yönelik veri tabanı tasarımını hazırlaması ve bunu dokümante edebilmesidir. Ayrıca tanımlanan veri tabanı tasarlama sürecinde gelecekte karşılaşılabilecek riskleri de öngörerek bu süreci gerçekleştirebilmek gerekmektedir.

Bu bölüm içerisinde doğrudan bir konu anlatılmayacak, bunun yerine gerçek dünya örnekleri üzerinde veri tabanı tasarımları gerçekleştirme alıştırmaları yapacağız. Böylece dersin bu aşamasına kadar eldeki tüm bilgileri birleştirecek ve bunları kullanarak asıl amacımıza ulaşacak şekilde kullanacağız.

Bölüm içerisinde 10 farklı alıştırma atölyesi bulunmaktadır. Her bir örnek için bir senaryo ile karşılaşacaksınız. Senaryolar içerisinde elde edilmek istenilen veri tabanıyla ilgili bazı ayrıntılar yer alacak olmakla birlikte bazı ayrıntılar kesin bilgiler içermeyecek. Böylece veri tabanı tasarımcısı olarak kesin olmayan durumlarda da gerçekleştirmeniz gereken yaklaşımı öğrenmiş olacaksınız.

Bu bölümde her bir problemi inceledikten sonra önce kendiniz çözüm üretmeniz, sonrasında çözüm yorumlarını incelemeniz oldukça önemlidir. Bu bölümle birlikte yazılım geliştirme projelerinde veri tabanı tasarımcısı olarak görev alma becerisine sahip olacaksınız.

8.1. Diyetisyenler İçin Platform

8.1.1. Problem

Bir diyet kliniği sahibi sizinle yeni kurgulanacak bir sistem için destek alma konusunda görüşmek istedi. Klinik sahibi, klinikte çalışan 8 diyetisyen olduğunu, destek almak isteyen kişilerin klinik sekreterlerini arayarak randevu aldığını ve görüşmelerin klinikte gerçekleştiğini belirtti. Yakın zamanda bu görüşmeleri internet ortamında da gerçekleştirmek istedikleri bir girişim içerisinde olduklarından bahsetti. Sizden bu girişim için gerçekleştirilecek bu sistemin veritabanının tasarlanması konusunda destek isteniyor. Yapılacak işi netleştirmek adına klinik sahibine çeşitli sorular yönelttiniz ve yanıtları not aldınız. Bu notlara istinaden bir veritabanı tasarımı gerçekleştiriniz. Aldığınız (aşağıda listelenen) önemli noktaların tamamına uyum sağlamalısınız. Sormayı ve not almayı unuttuğunuz durumlar için maalesef ki tekrar iletişime geçme şansınız yok. Dolayısıyla çeşitli varsayım ve öngörülerde bulunarak eksik gördüğünüz kısımları tamamlamalısınız.

Aldığınız notlar:

1) Sistem hem fiziksel hem de çevrimiçi ortamdaki rezervasyonların kayıtlarını tutmalı.

2) Birden fazla diyetisyen ile hizmet verebilmeli.

3) Sistemde yönetici, diyetisyen ve kullanıcı olarak üç yetki seviyesi yer almalı.

4) Gerçekleşmiş ya da gerçekleşecek tüm görüşmelerin başlangıç, bitiş saatleri ve diyetisyen ile kullanıcının katılım durumları kayıt altına alınmalıdır.

5) Kullanıcıların diyetisyenlerle görüşme yapabilmek için satın aldıkları paketlerin kayıt altına alınması gerekmektedir.

6) Her bir paket diyetisyene özeldir ve görüşme sayısı limiti içermelidir.

7) Bu durum için veritabanı tasarımını gerçekleştiriniz.

8.1.2. Çözüm Önerisi

Bir süre sonra tasarlama aşamasındaki birçok adımı doğallıkla gerçekleştirebilecek olsanız da bu işe ilk başladığınızda adımlarınızı temkinli atmanızda büyük fayda var. Problem başlığı altındaki anlatılar ve aldığınız farz edilen notlar, tasarım aşaması için büyük bir yol göstericidir. Hatırlarsanız önceki başlıklardan biri altında ad avına çıkmıştık. Nereden başlayacağınızı bilemediğiniz durumlarda sistemi özetleyen metin üzerindeki tüm adların bir listesini çıkararak gözden geçirin. Böyle bir metin yoksa siz yazın. Metnin dil ya da etik kurallara uyumlu olmasına gerek yok. Tüm özellikleri içerecek kadar detaylı ve mümkün olan en özet haliyle hazırlanması bu işlem için yeterli olacaktır.

Problem başlığı altında verilen metindeki adların bir listesini çıkaralım: Klinik, sistem, diyetisyen, kişi, sekreter, randevu, görüşme, fiziksel, çevrimiçi, rezervasyon, yönetici, kullanıcı, yetki, paket.

Şimdi bu adları irdeleyerek varlıklarımızı tanımlamaya başlayacağız. Sistemler genellikle kullanıcılar ve bunların rolleriyle ilgili varlıklar içerirler. Dolayısıyla öncelikle sistem kullanıcılarıyla ilgili kelimeleri seçip yorumlayarak başlayabiliriz. Kişi kelimesi kullanıcıyı kastetmekle birlikte diyetisyen, yönetici, sekreter ve kullanıcı kelimeleri kullanıcıların yetki seviyelerini belirtmektedir. Özel durumlarda açıklandığı gibi biz burada Kisi ya da Kullanici adında bir varlık tanımlayabilir ve tüm yetkileri bu varlık altında toplayabiliriz. İhtiyaç halinde daha sonra güncellemek üzere şimdilik bir varlık önerisinde bulunalım.

Kullanici: id, ad, soyad, yonetici, diyetisyen, sekreter

Kullanici varlığına, ihtiyaç duyulacağı öngörüsüyle id, ad ve soyad nitelikleri eklenmiştir. Bu varlığa ekli bir kayıt dolayısıyla bir kullanıcıdır. Kullanıcı, eğer yonetici niteliği 1 değeri alırsa yönetici, diyetisyen niteliğinin 1 değeri alması durumunda ise diyetisyen yetkisine sahip olacaktır. Benzer şekilde sekreter niteliği de 1 değeri almasıyla birlikte kullanıcıya bu yetkiyi kazandırır.

Kullanıcı ve yetki işini hallettikten sonra listemizdeki adları sırayla ele alabiliriz. Klinik ve sistem kelimelerini es geçiyorum. Bu kelimeler sistemin bütününe atıfta bulunan kelimeler. Bir varlık adı olmalarına ihtiyaç duymuyoruz. Randevu, görüşme ve rezervasyon kelimelerini de birlikte inceleyebiliriz. Bu kelimeler, bir diyetisyen ile kullanıcının yapacağı görüşmeyi referans alan kelimelerdir. Gerçekleşmiş ya da gerçekleşecek tüm görüşmelerin kayıt altında olması isteğinden bahsedilmişti. Burada görüşme kelimesini seçerek bir varlık tanımlayabilir, özellikleri bu varlığa bağlı şekilde belirleyebiliriz. Önerim aşağıdaki gibidir.

Gorusme: id, diyetisyen, kullanici, sekreter, gorusmeZamani, diyetisyenKatilimi, kullaniciKatilimi, ortam

Gorusme varlığı içerisinde bir ayrıntı dikkatinizi çekmiş olmalı. diyetisyen, kullanici ve sekreter nitelikleri birer kullanıcıyı işaret eden nitelikler. Tahmin edeceğiniz üzere bunlar ikincil anahtar olarak görev yapacaklar. Burada bir tablonun (Kullanici) kendi kendisiyle çoğa çok bağlantılı olmasına bir örnek görmekteyiz. Lütfen bu ayrıntıyı tasarımın tamamı üzerinde inceleyiniz.

Bu varlıkta görüşmenin tarafları olan diyetisyen ve kullanicinın yanı sıra bu kaydı oluşturmuş olması muhtemel olan sekreter de bulunmaktadır. gorusmeZamani niteliği görüşmenin gerçekleşeceği zamanı

içermektedir. Bu niteliğin bir zaman damgası olarak değer aldığını varsayabiliriz. diyetisyenKatilimi ve kullaniciKatilimi ise 0 ya da 1 değeri alan (boolean) ve bu kullanıcıların görüşmeye katılıp katılmadığı bilgisini içeren nitelikler olacaktır. Burada gorusmeZamaninin, içinde bulunulan andan önce ya da sonra olması görüşmenin geçmiş, gerçekleşmekte olan ya da gelecek olacak bir görüşme olduğu anlamını taşıyabilir. Bunun yanında diyetisyenKatilimi ve kullaniciKatilimi bu kullanıcıların görüşmeye katılım durumlarını içerdikleri için, geçmişte kalan ancak katılım gerçekleşmeyerek tamamlanamamış bir görüşmenin de kayıt altına alınabilmesini mümkün hale getirmektedir.

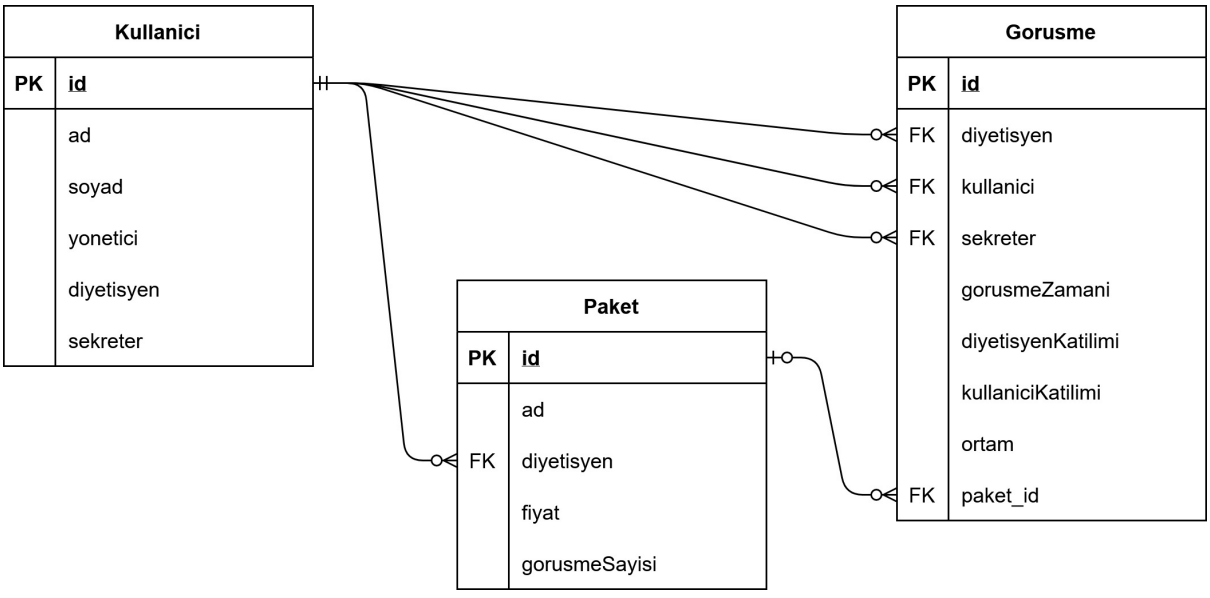
Son nitelik olan ortam, seçtiğimiz fiziksel ve elektronik kelimeleri için eklenmiştir. Bu kelimeler görüşmenin yapılacağı ortamın türünü belirtmektedir. Dolayısıyla yapılacak görüşmenin bir özelliği ve Gorusme tablosunun bir niteliğidir. ortam niteliği elektronik ya da fiziksel değerlerinden birini alabilir. Eğer burada saklama kapasitesinden tasarruf edilmek isteniyorsa kısaca e ve f harfleri de kullanılabilir. Bunun ötesinde bir diğer seçenek ortam niteliği yerine fiziksel ya da cevrimici adlı bir nitelik kullanılabilir. Örneğin, fiziksel niteliği kullanılması durumunda, 1 değeri görüşmenin fiziksel ortamda gerçekleştirileceğini; 0 değeri ise çevrimiçi ortamda gerçekleştirileceğini gösterecektir. Bu yaklaşımların tümü işe yarayacaktır.

Son olarak paket kelimesini ele alalım. Açıklamalar sayesinde her diyetisyen için birden fazla paket tanımlanabileceğini görmekteyiz. Öncelikle Paket adında bir varlık tanımlamalı, sonrasında da öngördüğümüz nitelikleri eklemeliyiz. Aşağıda bu varlık için önerimi sunuyorum.

Paket: id, ad, diyetisyen, fiyat, gorusmeSayisi

Paket varlığı içerisinde id ve ad nitelikleri paketi tanımlamayı sağlayan niteliklerdir. diyetisyen niteliği paketin hangi diyetisyene ait olduğunu gösteren bir ikincil anahtardır. fiyat niteliği paketin satın alma fiyatını, gorusmeSayisi ise bu paket satın alındığında gerçekleştirilebilecek görüşme sayısını içermektedir.

Tüm bu incelemeler sonucunda veri tabanı tasarımımız için bir öneri çizebilecek durumdayız. Önerilerimle aynı doğrultuda çizdiğim tasarım Şekil 8.1'de sunulduğu gibidir.



Üzerinde tartıştıklarımıza ek olarak burada Gorusme varlığı içerisinde paket_id niteliği göreceksiniz. Bunun sebebi Gorusme ve Paket varlıkları arasındaki olası bağlantıdır. Görüşme, satın alınmış bir paket dahilinde gerçekleştiriliyor olabilir. O halde bu görüşmenin paket ile ilişkilendirilmesi normaldir.

8.2. Araç Satışı Portalı

8.2.1. Problem

Kullanılmış arabaları alan ve satan bir işletme bir bilgi sistemine ihtiyaç duyarak sizinle iletişime geçti. İşletme sorumlusu, her türlü marka ve model aracın alınıp satıldığından bahsetti ve bunların kayıt altına alınmasını istediklerini belirtti. Bazı durumlarda eski müşterilerine avantajlar sunabilmek için geriye dönük

de sorgulamalar yapmak istediklerinden bahsedildi. Sistemin genel hatlarını öğrendikten sonra aşağıdaki maddeleri not aldınız.

1) Her bir araç birbirinden farklı olduğu için aynı marka ya da model olsa bile yeni kayıt girilmeli. (İkinci el ürün pazarı gibi)

2) Firma tüm marka ve modelleri alıp satıyor.

3) Alış ve satış kayıtları birbirinden bağımsız kayıt altına alınmalı.

4) Araç alış ya da satış işlemiyle ilgilenen müşteri temsilcisi de kayıt altında olsun.

5) Alış ve satış işlemlerinde yapılan ya da alınan tüm ödemeler kayıt altına alınmalı.

6) Ödemeler nakit, kredi kartı ya da banka havalesi ile gerçekleştirilebilmeli.

Bahsedilen sistem için bir veritabanı tasarımı gerçekleştiriniz.

8.2.2. Çözüm Önerisi

Çözüme başlarken problemin tarifinde ve ayrıntı içeren notlardaki olguların neler olduğunu belirlemeye çalışalım. Tüm problem başlığını dikkate aldığımızda sistemin aşağıdaki bileşenlere sahip olduğunu düşünüyorum.

Araba, İşletme, Marka, Model, Müşteri, Firma, Müşteri Temsilcisi,

Nakit, Kredi Kartı, Banka Havalesi

Bunun yanında aşağıdaki işlevlerden bahsedildiğini de görmekteyiz.

Alış, Satış, Kayıt, Ödeme

Olgu ve işlevlerin tamamını bir defada belirleyip, sonrasında türleri üzerine yoğunlaşmak da ele alınabilecek bir yöntemdir. Bu süreci tekrarladıkça olguları tanımanız ve nasıl kullanacağınız konusundaki bakış açınız daha da netleşecektir. Burada daha çok ad olarak geçen olguların birer varlık adı olmaya, işlev adlarının ise muhtemelen çoğa çok bağlantının varlığı olmaya aday olduğunu hatırlamak gereklidir. Şimdi ele aldığımız tüm kelimelerden hangilerinin işe yarayacağını belirlemeye çalışalım.

İşletme ve Firma aynı yapıyı ifade ediyor olmakla birlikte sistemin tamamını yöneten yapıda olduğu için bir varlık olarak tanımlama ihtiyacımız bulunmamaktadır. İşletmenin şubeleri ya da bölgeleri gibi bir şekilde ilişkili bileşenleri olsaydı bu konuyu biraz daha farklı değerlendirebilirdik. Ancak şu durumda işletme bilgisi bizim için herhangi bir varlık ile doğrudan bağlantılı değil, tüm sistemi kapsayıcı niteliktedir. Bu sebeple bu iki kelimeyi değerlendirme dışı bırakıyoruz.

Araba, probleme konu olan ve alış ve satışı yapılan varlığın ta kendisi olduğundan bir varlık olarak tanımlanması yerinde olacaktır. Bununla birlikte Marka ve Model kelimeleri, Araba varlığının bir özelliğini temsil ettiği için bu varlık altında nitelik olarak yapılandırılmaları uygun olacaktır.

Müşteri, alış ve satış işlerindeki muhatap kişi olmakla birlikte Müşteri Temsilcisi de bu işlemleri gerçekleştirmek üzere görevli işletme yetkilisidir. İki ifade de birer kişiyi temsil etmekle birlikte bir kişinin müşteri ya da müşteri temsilcisi sıfatı kazanması, o kişi için nitelik olarak görülebilir. Dolayısıyla bu iki olgu Kişi ya da Kullanıcı olarak birleştirilebilir ve görevi bir nitelik olarak ele alınabilir.

Nakit, Kredi Kartı ve Banka Havalesi olguları Ödeme işleminin bir özelliğidir. Bu sebeple bu üç olguya ödemeyle ilgili bir varlık içerisinde nitelik olarak yer vermek daha mantıklı olacaktır.

Sistemi işlevler açısından incelediğimizde temelde tek bir işlem olduğunu görmekteyiz: Araç satışı. Alış ve Satış işlemleri aynı anlama gelmekte, sadece satışın yönü değişmektedir. Kayıt işlevi de alış ve satış işlemlerin kaydı olduğu için ayrıca belirtilmesine gerek bulunmamaktadır. Ödeme işlevi ayrı ve tek başına

varlık olmayı hak eden bir varlık gibi gözükse de aslında alış ve satış işlemlerinden türeyen bir işlev olduğunu unutmamak gerekmektedir. Dolayısıyla Ödeme yeni bir varlık olarak yapılandırılacak ancak mutlaka satış işleviyle ilişkilendirilecektir.

Yaptığımız analizin sonuçlarını toplarsak neticede aşağıdaki varlıkları elde ederiz. Ayrıca bu varlıklar için bazı nitelik önerilerini de aşağıda bulacaksınız.

Araba: id, marka, model, plaka, motorGucu, yakitTuru

Kisi: id, ad, soyad, telefon, ePosta, yetki

Odeme: id, tutar

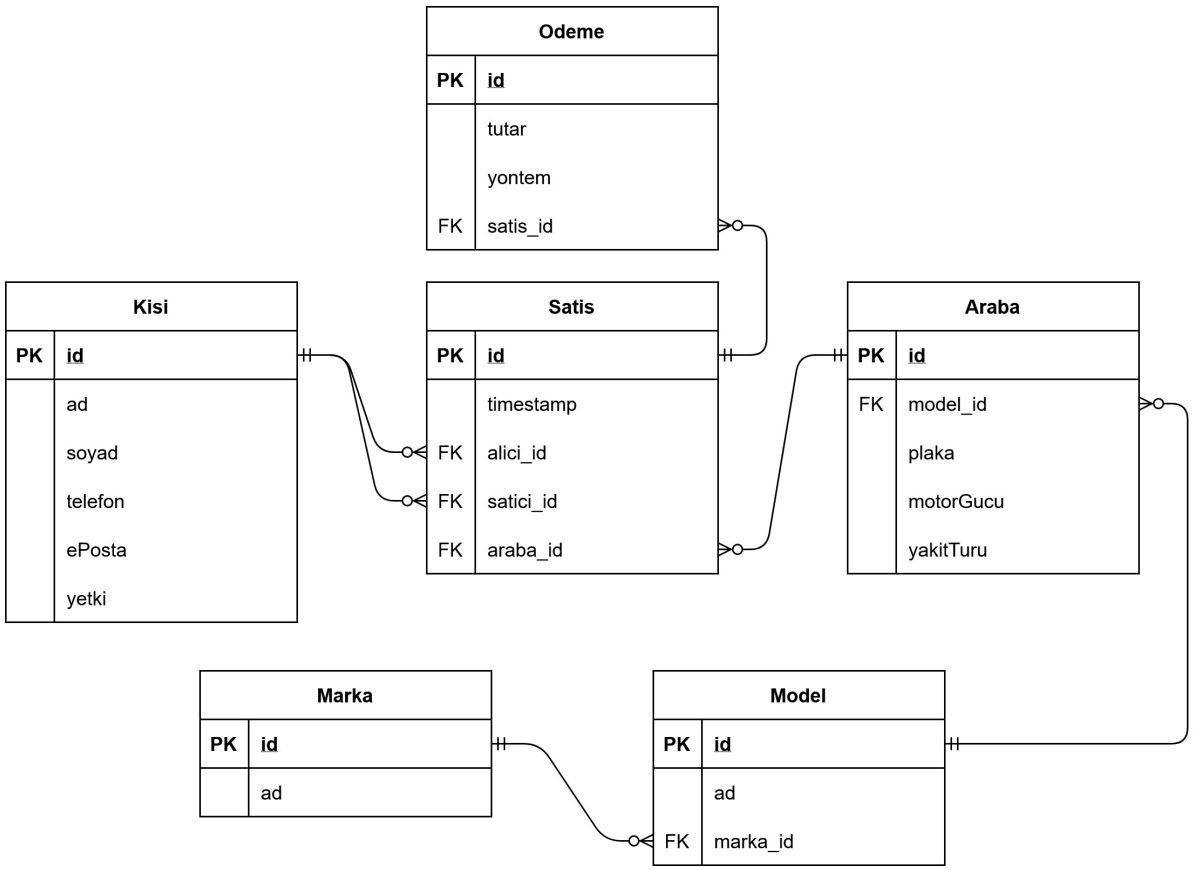
Satis: id, timestamp

Varlık önerilerini sunarken Odeme ve Satis varlıklarının birer işlevi temsil ettiğinden ve bunların çoğa çok bağlantı yapısındaki bağlantı varlığı özelliği taşıyor olmasının muhtemel olduğundan bahsetmiştik. Yukarıdaki varlık tanımlaması önerilerine dikkat ederseniz bu iki varlığın diğer iki varlığa ait bilgiler içermesi gerektiğini, dolayısıyla bir ilişki kurulması gerektiğini fark edebilirsiniz. Örneğin Satis varlığı alıcı ve satıcıya ait bilgiler içermelidir. Ayrıca ödemeye ait de bilgiler içermesi gerekmektedir. Benzer şekilde Odeme varlığı da bir satış ve bu satışta yer alan kişilerle ilişkili olmalıdır.

Araba ve Kisi varlıklarını ele alalım. Bu iki varlık birbiriyle ilişkili olmalıdır. Bu ilişki, işlevsel olarak satış işlemi olarak yorumlanabilir. Kontrol etmek açısından ilişki türü belirlemeyle ilgili kontrol sorularımızı yöneltebiliriz. Bir araba birden fazla kişiyle ilişkili olabilir. Bir kez satış işlemi olsa bile bir alıcı bir satıcı olduğu için iki kişiyle bağlantılıdır. Bunun yanında birden fazla kez satış işlemi de gerçekleşebilir. Benzer şekilde kişi de birden fazla araba ile ilişkili olabilir. Böylece bu iki varlığın çoğa çok ilişkili olduğunu teyit etmiş olduk. Aradaki ilişkinin de satış işlemi olduğunu bildiğimiz için bağlantı varlığı olarak doğrudan Satis varlığını kullanabiliriz.

Ödeme işleminin de bir satış ile bağlantılı olması gerektiğinden bahsetmiştik. Aradaki ilişki ise bizim bir varsayımda bulunmamızı gerektiriyor. Bir ödeme işleminin tek bir satış işlemiyle ilgili olduğunu varsaymak kolay olacaktır. Ancak bir satış işleminin tek bir ödemeyle ilişkili olduğunu söylemek o kadar da kolay değildir. Eğer her satış işleminin tek türde tek bir ödemeyle yapıldığını varsayarsak buradaki ilişki türü bire bir olacaktır. Ancak parçalı ödeme gibi yöntemlerin kullanıldığını düşünüyorsak bu durumda bir satış işlemi birçok ödemeyle ilişkilendirilebilir. Bu durumda da bire çok ilişki ile karşı karşıya kalacağız. Ödeme işlemleri genellikle hassas olduğu için varsayımlarımızı mümkün olduğunca herhangi bir hataya sebep olmayacak şekilde belirlememiz gerekmektedir. Bire bir bağlantı bizim için aynı zamanda bir kısıt sayılacaktır çünkü bu durumda bir satış işlemi ancak tek bir ödeme ile ilişkilendirilebilecektir. Bire çok bağlantıda ise bir satış işlemiyle birden fazla ödemeyi ilişkilendirebileceğimiz gibi tek bir ödemeyle de ilişkilendirebiliriz. Burada bire çok ilişki, bire bir ilişkinin bir kapsayanı olarak görev yapacaktır. Böylece veri yapısını olası karar değişikliğinde bile bozulmayacak olarak yapılandırmış olacağız. Bu sebeple Satis ve Odeme varlıkları arasındaki bağlantıyı bire çok türünde tanımlayabiliriz.

Tüm bu yorumlar neticesinde bir veri tabanı tasarımı önerisinde bulunalım. Çözüm önerisi için lütfen Şekil 8.2’yi inceleyiniz.



Şekilde, değerlendirmelerimizi yaparken konuştuğumuz varlıklara ek 2 varlık daha yer aldığını göreceksiniz. Bu varlıklar, Araba varlığının içerisindeki marka ve model nitelikleri tekrarlı veri içerecek olması sebebiyle eklendiler. Araba markaları belirli sayıdadır ve kendini tekrar edecektir. Benzer şekilde modeller de markalara bağlıdır ve onların değerleri de tekrarlıdır. Bu sebeple Marka ve Model iki varlık olarak eklenmiştir. Çözüm önerisinin başlarında bu iki olgu için Araba varlığının birer niteliği olduklarını ve ayrı varlık olmayı hak etmediklerini söylemiştik. Aslında görüşümüz hala geçerli. Veri yapısını incelediğimiz zaman bu iki olgu bizim için hala Araba varlığının nitelikleri. Burada sadece veri tekrarı olmaması ve daha etkin bir tasarım elde etmek amacıyla bu değerleri bir varlık içerisinde topladık ve Araba varlığına bir ikincil anahtar ekleyerek bağlantıyı sağladık. Bu da dolaylı da olsa hala bu iki özelliğin Araba varlığı içerisinde yer aldığını göstermektedir.

Çözüm önerisini incelediğinizde bir ayrıntının daha dikkatinizi çekmesi gerekiyor. Şekil 8.2 ile verilen veri tabanı tasarımı, bir önceki bölümde bahsettiğimiz özel durumlardan birine uyuyor. Lütfen, bu ayrıntı hakkında bir fikriniz yoksa tasarıma geri dönerek hangi özel duruma uyduğuyla ilgili inceleme ve tahminde bulunmaya çalışınız.

Yanıttan önce şunu söylemek gerekmektedir. Veri tabanı tasarımları hiçbir zaman birebir aynı şablon ile karşımıza çıkmazlar. Özel durumlar başlığı altında gördüğünüz durumlar, bu özelliğe sahip her bir veri tabanı için birebir aynı biçimde karşınıza çıkabileceği anlamı taşımamaktadır. Verilen tasarım, yıldız şema olarak değerlendirilebilecek bir tasarımdır. Dikkatli incelediğinizde bu tasarımda bir veri akışının kayıt altına alındığını görebilirsiniz. Satis varlığı, bahsettiğimiz veri akışının kayıt altına alındığı varlıktır. Kisi ve Araba varlıkları bu tasarımda boyut varlığı olarak değerlendirilebilir. Ödeme varlığı da bir boyut varlığı olmakla birlikte Kayıt tablosu kadar, belki daha fazla kayıt içeriyor olması sebebiyle ilgi çekici bir ayrıntı sunar. Marka ve Model varlıkları ise Araba varlığındaki veri tekrarını aşmak için uygulanmış bir çözümün neticesidir.

8.3. Para Transferi Yazılımı

8.3.1. Problem

Bir bankanın yazılım geliştirme departmanını çalışanısınız. Araştırma - geliştirme projeleri kapsamında, banka müşterisi olsun ya da olmasın herkesin birbirine para göndermesini mümkün hale getirebilecek bir

yapı üzerine çalışıyorsunuz. Bu yapıya göre kişiler telefon numaraları tarafından temsil edilmekte ve buna göre birbirlerine para gönderebilmektedir. Telefon numarası bilgisi kişilerin kimliği yerine geçmekle birlikte aynı zamanda doğrulayıcı rolü de üstlenmektedir. Bu projede veri tabanı geliştiricisi olarak yer almaktasınız. Gerçekleştirilen çok sayıda toplantı neticesinde özet olarak aşağıdaki notları aldınız.

1) Sistem kullanıcılarının banka müşterisi olma zorunluluğu yok.

2) Telefon numarası kişiler için kimlik niteliğinde.

3) Telefon numarası mutlaka SMS kodu yöntemiyle doğrulanmalı.

4) Gönderilecek para limiti veri tabanı üzerinde tutulmalı ve gerektiğinde güncellenebilmeli.

5) Para göndermek bir zaman aşımına sahip olmalı. Zaman aşımına uğrama durumunda göndericiye iade gerçekleşmeli.

Bu proje için bir veri tabanı tasarımı gerçekleştiriniz.

8.3.2. Çözüm Önerisi

Problemi analiz ederken öncelikle varlık adayı olgularımızı ve işlevlerimizi belirlemeye çalışalım. Problem başlığı altındaki ad özelliğindeki kelimeleri toparladığımızda aşağıdaki listeyi elde ediyoruz.

Banka, Müşteri, Telefon Numarası, Kişi, Doğrulayıcı, Kullanıcı,

SMS Kodu

Bunun yanında yine problem başlığı altında sistemin işlevi olarak sayılabilecek kelimeler de aşağıda listelenmektedir.

Para Göndermek, SMS Doğrulaması, Para Limiti, Zaman Aşımı, İade

Birçok sistemde farklı kullanıcı seviyeleriyle ilgili kayıtlar, daha önce bahsettiğimiz sebeplerden dolayı tek bir Kisi ya da Kullanici varlığı altında toplanırlar. Bu tür varlıklar her zaman bir kişiyi işaret etmekle birlikte sistemlerdeki farklı kullanıcı tipleri bu kişi varlığında bir nitelik olarak kolaylıkla tutulabilir ve veri bütünlüğünü mümkün hale getirebilirler. Ayrıca tüm kişileri tek bir varlık içerisinde toparlamak, gerçekleşmesi olası yetki değişikliklerinde bir kaydın varlıktan varlığa taşınmasındansa bir kaydın tek bir niteliğinin güncellenmesi ile sağlanabilmektedir. Bu sebeplerden dolayı gerçekleştireceğiniz veritabanı tasarımlarında eğer kullanıcılar işin içindeyse, her ne kadar farklı yetki seviyeleri ya da kullanım şekli olsa da kullanıcılara ait tek bir varlık iş görecektir. Bu çözümün riskli olduğunu düşündüğünüz durumlarda elbette farklı bir çözüm uygulayabilirsiniz ancak rahatlıkla söyleyebilirim ki bu yöntem yüzde 90'ın üzerinde iş görecektir.

Bu açıklama üzerine, sistemimizin kullanıcıları olduğu gerçeğini hatırlayarak bu tabloyu tanımlamak üzere hangi olguları listeden elediğimize bir bakabiliriz. Musteri, Kisi ve Kullanici yerine ortak bir ad seçerek devam edelim: Kullanici. Banka, sistemin tamamına ait bir tanımlama olduğu için bir varlık olarak belirtmeye ihtiyacımız yok. Eğer problem açıklaması içerisinde Bir bankanın ... şeklinde açıklama yapılması yerine birden fazla bankanın ortak gerçekleştirdiği bir süreç olarak tanımlansaydı elimizde birden fazla banka olacağı için Banka varlığını kullanmak gerekli olacaktı. Telefon Numarasi, Kullanici niteliğine ait bir özellik gibi görünüyor. Ancak bu konuda biraz daha ayrıntı var. Banka kullanıcılarının, bankanın müşterisi olmayan kişilere telefon numarası üzerinden para gönderilebileceğinden bahsettik. Dolayısıyla para gönderimi yapılacak telefon numarası hali hazırda bir tablo içerisinde kayıtlı olmayabilir. Sistemde kayıtlı olmayan bir kişiye para gönderimi yapabileceğimiz için Telefon Numarasi bir kullanıcının değil ödeme işleminin bir niteliği olmalı. Bu durumu tasarımı gerçekleştirdikten sonra ayrıntılı yorumlayacağız. Dogrulayici, kişinin telefon numarasının kendisine ait olduğunu kanıtlaması için kullanılan, banka tarafından oluşturulan ve telefon numarasına gönderilen özel bir kodu ifade etmektedir. Kişi, telefon numarası gerçekten kendisine aitse gelen kodu bankaya geri bildirir ve numaranın kendisine ait olduğunu ispatlar. Bu kod, ödeme işleminin bir niteliği olmalıdır. Her bir ödeme işleminde yeni bir kod üretilmeli ve ödeme kaydıyla birlikte saklanmalıdır.

Şimdi işlevleri inceleyelim. Para gönderimi, iki kişi arasında gerçekleştirilen bir işlemdir. Kişi varlığının kendi kendisiyle çoğa çok bağlanması sonucunda oluşabilecek bağlantı varlığı olarak da karşımıza çıkmaktadır. Bu varlığa Odeme adını verelim. Bir kişi birden fazla kişiye para gönderebilirken, bir kişi birden fazla kişiden para alabilir. Ancak burada alıcının Kullanici tablosunda kayıt olma zorunluluğu yoktur. Tasarımı gerçekleştirirken bu ayrıntıyı tasarım üzerinde mutlaka belirtmemiz gerekmektedir. SMS doğrulamasının Odeme varlığında yer alması gereken bir nitelik olduğundan bahsetmiştik. Para limiti ve zaman aşımı değerleri sisteme ait bir ayar varlığında saklanmalı, bunun yanında zaman aşımı her bir ödeme için de kayıt altına alınmalıdır. Ayar varlığı zaman aşımının ne kadar süreceğini kayıt altında tutmalı, Odeme varlığı ise kayda alınan ödemenin hangi zamana kadar geçerli olduğu bilgisini bulundurmalıdır. İade işlemi yeni bir varlık yerine mevcut ödeme kaydı üzerinde nitelik ile tutulabilecek bir bilgidir. Yeni bir varlık olarak tanımlamak hata olmayacağı gibi bu konuda ek bilgilerin kayıt altına alınması istenilmediği için mevcut varlık içerisinde bu bilgiyi tutmak çözümü pratikleştirecektir.

Bu bilgilere göre varlık tanımlarımız için aşağıdaki planlamayı kullanabiliriz.

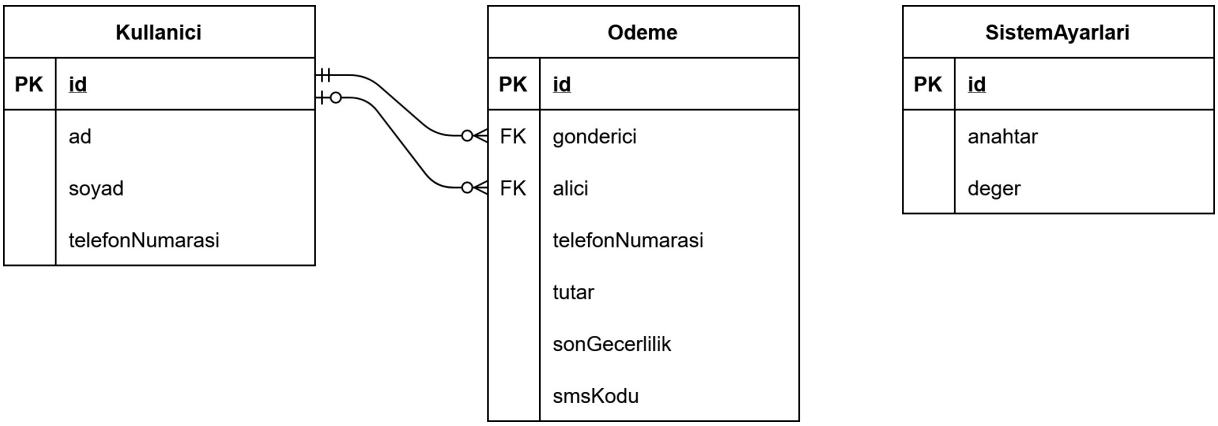
Kullanici: id, ad, soyad, telefonNumarasi

Odeme: id, gonderici, alici, telefonNumarasi, tutar,

sonGecerlilik, smsKodu

SistemAyarlari: id, anahtar, deger

Bu tanımlamalara ve yapılan açıklamalara uygun olarak hazırlanan veri tabanı tasarımı Şekil 8.3 ile sunulmuştur.



Tasarım önerisi üzerindeki bazı ayrıntıların üzerinde duralım. Öncelikle bu tasarım da yıldız şemanın bir örneği olabilir. Her ne kadar tek boyuta sahip olsa da ödemeyle ilgili tüm kayıtları içeren bir kayıtsal veri yapısı ile karşı karşıyayız. Odeme varlığı, Kullanici varlığının kendi kendisiyle yaptığı çoğa çok bağlantının bağlantı varlığı olarak da görülebilir. Bir kullanıcının hiçbir ödeme yapmaması ya da almaması muhtemel olduğu için bağlantıların Odeme varlığı ile birleşim noktalarında zorunluluk olmaması normaldir. Ancak gönderici kullanıcı bağlantısında zorunluluk olduğuna, alıcı kullanıcıda ise herhangi bir zorunluluk olmadığına dikkatinizi çekmek isterim. Bunun sebebi problem içerisinde banka müşterisi olmayanlara da para gönderilebileceğinin belirtilmiş olmasıdır. Bu ayrıntı sebebiyle Odeme varlığı alici ikincil anahtarı üzerinden her zaman Kullanici varlığı ile ilişki kuramayabilir.

Bir diğer ayrıntı telefonNumarasi niteliğinin iki varlık içerisinde birden bulunması. Odeme varlığı hem alici hem de gonderici üzerinden Kullanici varlığı ile bağlantılı. Dolayısıyla telefon numarası bilgisi erişilebilir. Ancak burada ayrıntı şu ki; telefon numarası ödeme işlemi için büyük önem taşıyor. Özellikle alıcının telefon numarası. Az önce bahsettiğimiz üzere ödeme alıcısının Kullanici varlığında herhangi bir kaydı olmayabilir. Ancak bizim bu bilgiyi boş geçmemiz mümkün değil. Dolayısıyla telefon numarası bilgisinin mutlaka ödeme ile ilişkilendirilmesi gerekiyor. Eğer gönderici ve alıcılar Kullanici varlığı içerisinde tanımlı olsaydı bile ödeme yapılmak istenilen telefon numarası kullanıcının varsayılan iletişim hattından farklı olabilirdi. Bu durumda da ödeme kaydı oluşturulacak telefon numarasının farklı olarak kaydedilebilmesi gerekiyordu. Her halükarda bu çözüm riskleri en aza indirebiliyor. Unutmayın, veri bütünlüğünü sağlamak birincil önceliğimizdir.

Son olarak sistem ayarları ve zaman aşımı yönetimini ele alalım. Sistem ayarları, özel durumlarda da bahsettiğimiz üzere diğer varlıklardan ayrık bir şekilde veri tabanına kaydedilebilir. Burada istenilen anahtara bağlı değerler tutularak sistem ayarı olarak kullanılabilir ve ihtiyaç halinde güncellenebilir. Problemdeki taleplerden biri ödemelerin bir geçerlilik tarihine sahip olabilmesiydi. Bu bilgi SistemAyarlari varlığına aşağıdaki gibi bir kayıtla eklenebilir:

id: 1, anahtar: gecerlilik, deger: 86400

Geçerlilik süresi ayar varlığı içerisinde gecerlilik anahtarı ile saklanmış olmaktadır. Değeri ise görüldüğü üzere 86400'dür. Bu değer 1 güne karşılık gelen saniye sayısıdır. Bu kayıt sayesinde yeni bir ödeme kaydı oluşturulduğunda Odeme varlığındaki sonGecerlilik niteliğinin değeri şu şekilde belirlenebilir:

Odeme.sonGecerlilik = simdi() + 86400

Burada simdi() fonksiyonu şu ana ait zamanı timestamp cinsinden veren fonksiyon olsun. 86400 değeri ise SistemAyarlari varlığından çekilen güncel gecerlilik değeridir. Bu durumda ödemeye ait son geçerlilik, kaydın oluşturulduğu andan tam bir gün sonrasını işaret edecektir. SistemAyarlari varlığında yapılacak bir güncelleme yeni eklenecek kayıtlarda etki edecek ancak geçmiş kayıtlar için herhangi bir etkisi olmayacaktır.

8.4. Nakliye Firması Organizasyon Yazılımı

8.4.1. Problem

Bir kargo firması için kargo takip sistemi geliştirme ekibine dahil oldunuz. Projedeki göreviniz veri tabanı tasarımını gerçekleştirmek. Firma, Türkiye'deki tüm şehirlere hizmet veren bir kargo firmasıdır. Bir gönderi, kargo firmasının herhangi bir şubesinden yolculuğa başlar, çeşitli aktarma merkezlerinden geçerek varış şubesine ulaşır. Varış şubesi personelinin de adrese teslim etmesiyle sonlandırılır. Geliştirilecek projeyle ilgili tüm ayrıntıları konuştuktan sonra bazı notlar aldınız. Aldığınız notlar şu şekilde:

1) Kargonun tüm hareketlerinin dökümü alınabilmeli.

2) Sistem, müşteriler ve çalışanlar tarafından erişilebilir olmalıdır.

3) Müşteriler bilgileri yalnızca görüntüleyebilmeli, çalışanlar yeni kayıt ekleyebilmeliler.

4) Mevcut kayıtlar silinememelidir.

Taleplere uygun bir sistem geliştirilebilmesi için uygun veri tabanı tasarımını gerçekleştiriniz.

8.4.2. Çözüm Önerisi

Daha çözüm üzerine herhangi bir adım atmadan yıldız şema biçiminde bir çözüme ulaşacağımızı tahmin edebileceğimiz bir örnek ile karşı karşıyayız. Çoğumuzun aşina olduğu kargo hizmeti, bir malın iki kişi ve/veya kurum arasındaki nakliyesini konu almaktadır. Bu hizmet için oluşturulmuş bir sistem, sürekli yeni teslimatların sisteme eklenmesi ve tüm teslimatların süreçlerinin kayıt altında olması sebebiyle geriye dönük işlemlerin de kayıt altında bulunması neticesinde bir kayıtsal veri tabanı biçiminde değerlendirilebilecektir. Dolayısıyla kargo hareketlerini içeren bir varlığa sahip olacağımızı ve bunun yıldız şemadaki kayıt (fact) varlığına denk geleceğini savunabiliriz. Bunları öngörüyor olabilsek de herhangi bir hata yapmamak için izlememiz gereken adımları izleyelim.

Problem başlığı altındaki kelimeleri incelediğimizde ad özelliğine sahip olanların varlık, fiil olanların ise bağlantı varlığı olmaya aday olduğunu gördük. Bu noktada varsa sıfatların da genellikle bir nitelik adayı olduğunu belirtelim. Şimdi adları ve fiilleri seçerek aday varlıklarımızı bulmaya çalışalım.

Kargo, Gönderi, Aktarma Merkezi, Şube, Personel, Müşteri,

Çalışan

Bunlara ek olarak aşağıdaki ifadeleri fiiller olarak değerlendirebilir ve sistemin işlevleri olarak dikkate alabiliriz.

Takip, Aktarma, Teslim, Görüntülemek, Kayıt

İlk kelimeyle incelememize başlayalım. Kargo, bu sistemde takibi yapılacak ürünlerin kayıtlarının tutulması için gerekli olan bir varlıktır. Bununla birlikte listedeki Gonderi ifadesi de aynı amaçla kullanılmış; bir kargonun nakliye edilmesi sürecini kasteden bir kelimedir. Kargo adı sistemin geneli için de kullanılan bir kelime olduğundan burdaki varlık adını Gonderi olarak seçmeyi tercih edebiliriz.

AktarmaMerkezi gönderilerin nakliyeleri esnasında kullanılan merkez birimlerini, Sube ise gönderi nakliyatının başlangıç ve bitiş noktalarındaki ofisleri ifade etmektedir. Bu iki ifade birleştirilerek Birim adı altında tek bir varlıkta toplanabilir. Tüm nakliye süreci için gönderinin uğradığı noktalar Birim adıyla anılabilir.

Personel, Musteri ve Calisan ifadeleri temelde sistem kullanıcılarını ifade etmekle birlikte kullanıcının sahip olduğu yetkiye bağlı bir farklılık içermektedir. Önceki atölyelere benzer şekilde burada bir Kullanici varlığı tanımlayabilir, kullanıcının türünü bir nitelik ile kayıt altına alabiliriz.

Kelime çıkartırken ayırdığımız fiillerin bizim için sistemin işlevlerini temsil ettiğini söylemiştik. Şimdi bunları ele alalım. Takip, her bir gönderi için nakliye sırasındaki işlem kayıtlarının tutulduğu bir varlık olarak kurgulanabilir. Böylece gerçekten de gönderinin her bir adımının takip edilmesini sağlayan bir veri yapısı elde edebiliriz. Aktarma, yine gönderinin hareketleriyle ilgili bir eylem olduğu için Takip tarafından kapsanan bir özellik olacaktır. Dolayısıyla bu ad için yeni bir varlığa ihtiyaç duymayacağız. Benzer şekilde Kayıt da işlevsel olarak bu varlığın işlevi kastedilerek kullanıldığı için göz ardı edilebilir.

Teslim süreci, bir gönderinin hedefe ulaşmasıyla ilgili bilgidir. Dolayısıyla bu bilgi bir Gonderinin niteliği olmalıdır. Yeni bir varlık oluşturmak yerine Gonderi varlığına teslimle ilgili nitelik ya da nitelikler oluşturarak bu bilgiyi tutabiliriz. Bu durum karmaşık geldiyse test etmek üzere bir deneme yapabiliriz.

Gonderi varlığıyla birlikte Teslim adında da bir varlık oluştursaydık bunların arasındaki ilişki tipi ne olurdu?

Bir gönderi, yalnızca tek bir teslim bilgisine sahip olmalıdır. Bir teslim bilgisi de yalnızca tek bir gönderiye ait olmalıdır. Burada bire bir ilişki ile karşılaşıyoruz. Bire bir ilişkilerin genellikle ihtiyaç duyulan ya da tercih edilen bir ilişki türü olmadığından bahsetmiştik. Bu ilişki türü genellikle özellikle ihtiyaç duyulduğunda kullanılan, zorunluluk gerektirmeyen ilişki türleridir. Bir tasarımda bire bir ilişki ile karşılaştıysanız ve bunu planlayarak yapmadıysanız bir hata yapmış olmanız olasıdır. Bu sebeple ilişki türünü belirlerken dikkatli olmakta fayda var.

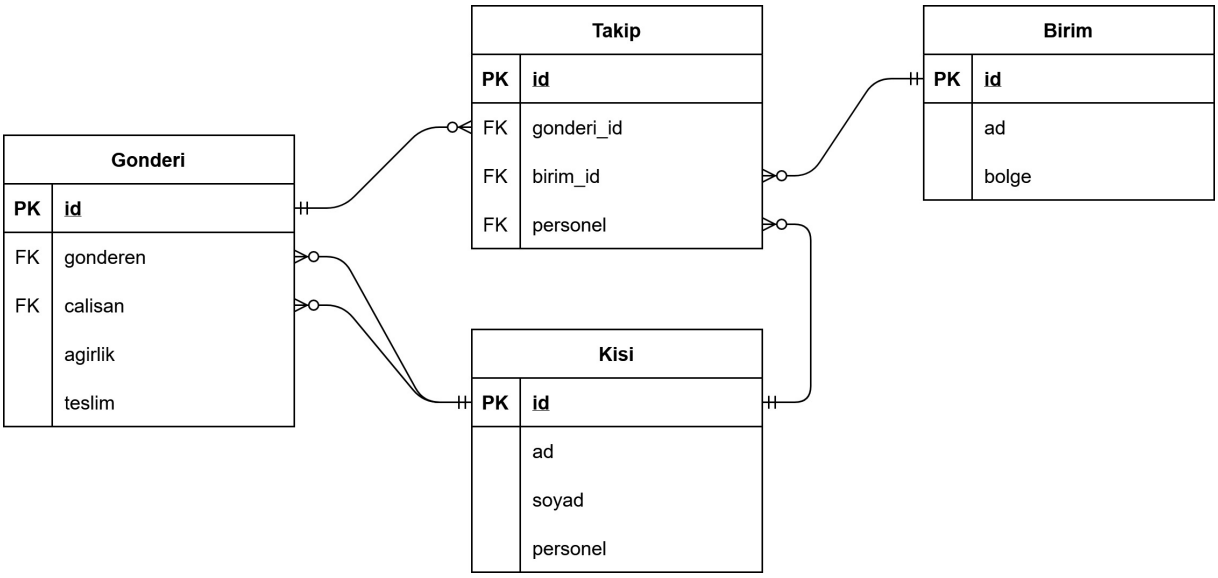
Peki burada bire bir ilişki kullanmak her zaman hata mıdır? Yanıt: Hayır. Eğer ki biz teslimat ile sadece teslim edildi ya da edilmedi bilgisini tutmuyor, bunun yerine teslimatla ilgili çok fazla nitelik barındırıyorsak Gonderi varlığını kullanmak istemeyebiliriz. Bu durumda yeni bir varlık oluşturarak Gonderi tarafında zorunluluk olan, Teslim tarafında olmayan bire bir bağlantı da çözüm olabilir. Ele aldığımız problemde böyle bir ayrıntı sunulmadığı için teslimat bilgisini tamamlandı ya da tamamlanmadı bilgisi içerecek şekilde kurgulayabilir, dolayısıyla Teslim varlığına ihtiyaç duymayabiliriz.

Son olarak görüntülemek kelimesini ele alalım. Sistemi talep eden kargo firması, bu işin uzmanı olmadıkları için hangi özelliği kimden talep edeceklerini bilemeyebilirler. Bilgilerin müşteriler tarafından yalnızca görüntülenmesini, personelin sadece giriş yapıp silme işlemi gerçekleştirememesini talep edebilir ve bunun veritabanı sorumlusu olduğunuz için sizin göreviniz olduğunu düşünebilirler. Ancak bu tarz arayüz kısıtlama işlemlerin genellikle yazılımın içerisinde gerçekleştirilmektedir. Dolayısıyla siz veri tabanında kullanıcının yetkisini tutarsınız, yazılım ve arayüz geliştiricileri de sağladığınız yetki bilgisine göre ilgili ekranları kullanıcıya gösterir.

Veri tabanları farklı kullanıcılar tarafından kullanılabilecek bir yapıya da sahip olsa da veri tabanı için tanımlanan kullanıcılar genellikle sistemin kendisidir. Geliştirilen bir sistem hem web tabanlı hem masaüstü hem de mobil uygulama olabilir. Bu durumda veri tabanı için 3 kullanıcı oluşturmak bazı avantajlar sağlayabilir. Ancak sistemi kullanan her bir kullanıcı için veri tabanı seviyesinde ve bu erişimi sağlayan bir kullanıcı tanımlanması pek de karşılaşılan bir uygulama değildir. Buradaki ayrımı oluşturan çizgi biraz

bulanık gibi görülebilir. Değerlendirmemize genel çerçeveden bakarsak şu yorum ile toparlayabiliriz: Veri tabanı veriyi saklar ve sunar. Bu verinin nasıl kullanılacağı yazılım içerisinde çözülmesi gereken bir problemdir.

Analiz ve yorumlamalar neticesinde ilgili problem için veri tabanı tasarımı Şekil 8.4 ile sunulmuştur.



8.5. Operatör Koordinasyon Yazılımı

8.5.1. Problem

Teknik tarafta meydana gelen gelişmeler sebebiyle bir GSM operatörünün altyapısının yenilenmesi kararı alındı. Yenileme çalışmalarını gerçekleştirecek ekip içerisindesiniz. Tüm fizibilite aşamasında ekip ile birlikte hareket ettiniz ve geliştirmeye yönelik ayrıntılı bilgiye sahipsiniz. Geliştirme sürecinde ilgili sistemin veri tabanı tasarımını hazırlamaktan sorumlu olacaksınız. Fizibilite aşamasında edindiğiniz bilgilere ait notlar aşağıdaki gibidir.

1) Tüm telefon hattı sahiplerinin kişisel bilgileri de kayıt altına alınmalı.

2) Hatlar faturalı ya da faturasız (bakiye yüklemeli) olabilir.

3) Hatlara belirli konuşma süresi, sms ve internet kotası tanımlı paketler atanabilir. Bu paketler belirli tarihler arasında aktifleştirilebilir olmalı.

4) Paketlerin kalan bakiyeyi sonraki aya devretme özelliği olabilir.

Bu duruma uygun veri tabanı tasarımını hazırlayınız.

8.5.2. Çözüm Önerisi

Kelimeleri analiz ederek çözüme başlayalım. Öncelikle varlık aday olan adları yakalamaya çalışalım. Önerilerim şu şekilde:

Telefon, Hat, Sahip, Kişisel Bilgi, Faturalı/Faturasız, Bakiye,

Konuşma, Sms, İnternet, Kota, Paket,

Fiil olan ve sistem için işlevsel özellik taşıma potansiyeline sahip kelimelere de bir göz atalım.

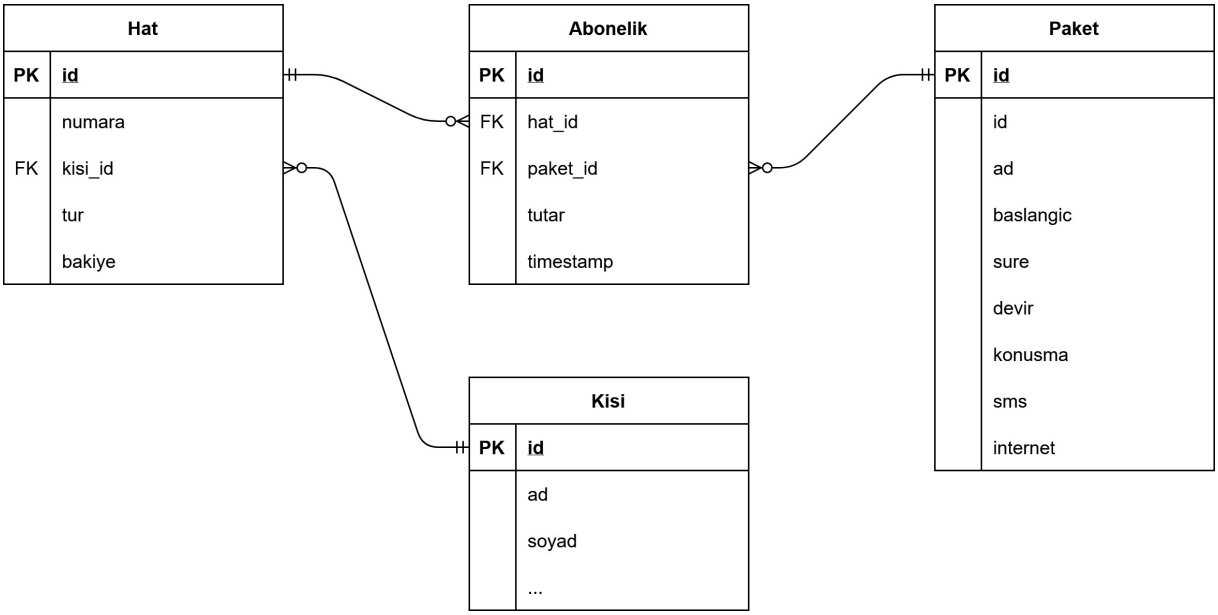
Paket Atama, Aktifleştirme, Devretme

Tasarlanan sistemler genellikle en az bir kullanıcıya sahip olacakları için benim gözüm öncelikle kullanıcı kelimesini arıyor. Farklı yetki seviyeleri için farklı kelimelerle anılan kullanıcılar sayesinde birden fazla kelimeyi bu havuzdan çıkararak kullanıcı kelimesi altında rahatlıkla birleştirebiliyorum. Örneğimizde sahip kelimesi hattın sahibini, dolayısıyla kullanıcının ta kendisini işaret etmektedir. Bununla birlikte bir de kişisel bilgi ifadesi mevcut. Kişisel bilgiler, kullanıcıyla ilgili varlık içerisinde bir araya getirilebilir. Ancak bir önceki atölyede ele aldığımız üzere bu tarz birçok nitelik içeren bir ek varlık ayrıca oluşturularak bire bir bağlantı ile de tasarlanabilir. İki yaklaşım da yanlış olmayacaktır. Kullanici varlığı oluşturup hem kullanıcı bilgileri hem de kullanıcının kişisel bilgilerini bu varlık içerisinde tutabiliriz. İkinci yaklaşımda ise Kullanici ve KisiselBilgi varlıklarını tanımlayabilir, bunları bire bir bağlantı ile kurgulayabiliriz.

Kullanici ve KisiselBilgi varlıkları arasında oluşturulacak bire bir bağlantı üzerinde zorunluluk durumları nasıl seçilmeli ve ikincil anahtar hangi varlık içerisine yerleştirilmelidir?

Telefon ve Hat kelimeleri neredeyse aynı olgu için kullanılıyor olsa da kurgu için benim önerim, Hat adlı bir varlık tanımlamak ve numara adlı bir niteliği bu varlık içerisine yerleştirmektir. Bir hattın faturalı ya da faturasız olduğu yine o hat için bir nitelik ile tutulabilecek bir bilgidir. Bu sebeple tur adlı bir niteliğin oluşturulması yerinde bir karar olacaktır. Bakiye bilgisi de yine hatta ait bir özelliktir. Bu bilgi de bakiye niteliği olarak Hat varlığına eklenebilir. Bunların yanında konuşma, sms ve internet kotalarının da hatta ait olduğu düşünülebilir olsa da bunlar aslında hat için abone olunan pakete ait bilgilerdir. Bu sebeple öncelikle Paket adında bir varlık tanımlanmalı; konusma, sms ve internet nitelikleri bu varlık içerisine yerleştirilmelidir.

İşlevleri belirleyebilmek için seçtiğimiz fiilleri ele aldığımızda, paket atama ifadesini bir paket için gerçekleştirilen abonelik olarak değerlendirilebilir ve Abonelik adlı bir varlık olarak tanımlayabiliriz. Paket atama ile kastedilen de abonelik durumudur. Devretmek ise bir paketle ilgili tanımlanabilecek bir özelliktir. Dolayısıyla Paket varlığına devir adında bir nitelik ekleyerek bu bilgiyi sağlayabiliriz.



Problem için önerilen veri tabanı tasarımı Şekil 8.5 ile sunulmuştur. Burada, Kisi varlığı içerisindeki .. ifadesi, kişisel bilgilerle ilgili niteliklerin burada yer alabileceğini ya da yeni bir tablo ile bu tabloya bağlayabileceğini ifade etmek için eklenmiştir. Kendi çözümünüzde bu iki yaklaşımdan birini uygulayabilirsiniz. Bunların yanında problemde belirtilmeyen ama bulunmasının gerekli olduğu tahmin edilebilecek bazı nitelikler de çözüme dahil edilmiştir.

8.6. Elektronik İletişim Yazılımı

8.6.1. Problem

Kişisel verilerin yabancı ülkelere gitmesini engellemek üzerine bir ulusal anlık mesajlaşma uygulaması geliştirme ekibi kuruldu ve bu ekipte veri tabanı uzmanı olarak yer almaktasınız. Geliştirilecek sisteme ait

ayrıntılar, aşağıda maddeler olarak sunulmuştur.

1) Her bir kullanıcı için telefon numarası yazılımdaki kimlik numarası olarak kullanılmalı.

2) İki kişi arasında özel mesajlaşma yapılabilecek.

3) İkiden fazla kişiyi içeren mesajlaşma grupları kurulabilecek.

4) Mesajlar; metin, görsel, video ya da ses kaydı biçimlerinde hazırlanabilir.

5) Mesajlar kullanıcılar tarafından silinse bile veri tabanında tutulmaya devam edilmeli.

Bu sistem için uygun bir veri tabanı tasarımı hazırlayınız.

8.6.2. Çözüm Önerisi

Oldukça popüler olan bu uygulama türü için elbette biz de bir veri tabanı tasarımı önerisi getirebiliriz. Problem başlığı altındaki kelimeleri incelediğimizde olgular için şu önerileri sunabiliriz:

Kullanici, Telefon Numarası, Kimlik Numarası, Mesaj, Özel

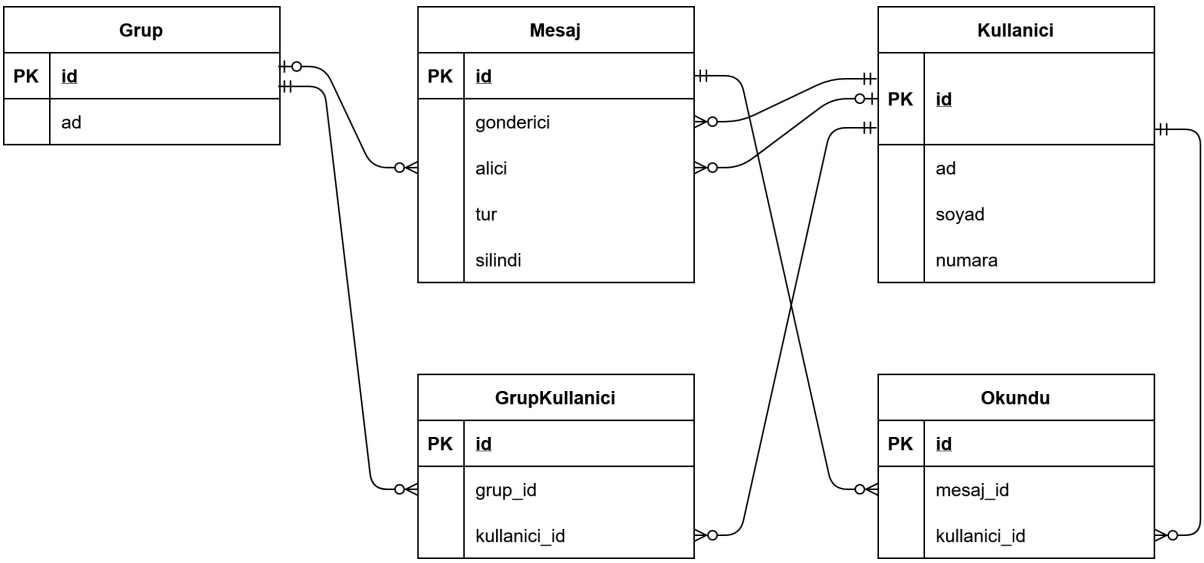
Mesajlaşma, Mesajlaşma Grubu, Metin, Görsel, Video, Ses

Sistemin işlevlerini tanımlayabilmek için ise aşağıda verilen, problem başlığı incelenerek çıkartılmış kelimeler fayda sağlayabilir.

Mesaj Gönderme, Silme, Okundu

Burada Kullanici varlığını hiç tartışmadan, doğrudan tanımlayabiliriz. Bununla birlikte numara niteliğini de bu varlığa eklemek gerekecektir. Kimlik Numarası ile kastedilen zaten bu telefon numarasının kullanıcı için ayırt edici olarak kullanılmasını sağlamaktı. Bu normalde birincil anahtar olarak kullanmak anlamına gelse de biz varlıkları tanımlarken her türlü riskten kaçınmak adına yalnızca id birincil anahtarını kullanacağımızdan bahsetmiştik. Telefon numarasının eşsiz bir kimlik bilgisi olarak tutulabilmesi yazılım katmanında sağlanabilecek bir durumdur. Bir mesajlaşma uygulaması için Mesaj adlı bir varlığı elbette tanımlanmalıdır. Bu varlık, sistem üzerinde gönderilen tüm mesajları saklayabilecek olan bir kayıt varlığı olarak da görülebilir. Özel mesajlaşma, mesajın iki kişi arasında gerçekleştirilebileceğini gösteren bir ifadedir. Tasarımımızı yapılandırırken bu ayrıntıya dikkat etmemiz gerekecektir. Mesajlaşma grubu özelliği için öncelikle grupların saklanması, sonrasında ise hangi kullanıcıların hangi gruplarda yer aldığının bilgisini saklamak üzerine birer varlığa ihtiyacımız bulunmaktadır. Bu sebeple Grup ve GrupKullanici varlıklarını oluşturabiliriz. Burada Grup varlığının oluşturulması kararı önemlidir. Ancak katılımcılarla ilgili ikinci tabloyu gözden kaçırmış olabilirsiniz. Eğer böyle olduysa bile tasarımı gerçekleştirirken Kullanici ve Grup arasında kurmak isteyeceğiniz ilişkinin çoğa çok olduğunu farkedecek, bir bağlantı varlığına ihtiyaç duyacaktınız. Bu bağlantı varlığı grup katılımı bilgisini içerecek olan varlıktır. Dolayısıyla ilk adımda kaçırılması muhtemel olsa bile tasarım tamamlanmadan eksikliği farkedilecek türden bir varlıktır. Metin, görsel, video ve ses ifadeleri mesajın türünü göstermektedir. Dolayısıyla Mesaj varlığına eklenecek bir tur niteliği bu bilginin saklanabilmesini sağlayacaktır.

İşlevlere geldiğimizde mesaj gönderme Mesaj varlığına kayıt eklendiği anda gerçekleştirilmiş olacak bir işlemdir. Dolayısıyla Mesaj varlığı bizim için hem bir olgu hem de kayıt tablosu olarak kullanılabilecek niteliktedir. Mesajların silinme bilgisi, eğer biz mesajı gerçekten silmek istemiyorsak, ilgili kayda ait bir silindi niteliği kullanılmasıyla tutulabilir. Bu sebeple Mesaj varlığına bir silindi niteliği eklemek bizim için iyi bir çözümdür. Okundu bilgisi için ise ele almamız gereken iki olası durum var. Birincisi özel (bire bir) mesajlaşma. Mesajlaşmanın sadece iki kişi arasında olduğu durumda, mesaj gönderici tarafından zaten okundu kabul edilebileceği için yalnızca alıcının okuyup okumadığı bilgisini kayıt altına almak yeterlidir. Ancak bir grup mesajlaşmasında mesajın okunma bilgisi çok kişi için tutulmalıdır. Bu da Mesaj ile Kullanici varlıkları arasında çoğa çok bağlantıya sebep olacak, dolayısıyla bir bağlantı varlığı gerekecektir. Okundu adı verebileceğimiz bu bağlantı varlığı sayesinde tüm kullanıcıların tüm mesajlar için okuma bilgisini kayıt altına alabiliriz. Bu yorumlara göre tasarım önerisi Şekil 8.6 ile sunulmuştur.



Burada Mesaj varlığı üzerinde biraz durmak gerektiğini düşünüyorum. Sistemin en önemli parçası olmakla birlikte sistemin kullanıcılar arasında ya da grup mesajlaşmasına müsade etmesini sağlamak; bu varlık üzerinde bir özel durumun oluşmasına sebep oldu. Mesaj.gonderici her zaman bir kullanıcıyı gösteren ikincil anahtardır. Çünkü bir mesaj her zaman bir kullanıcı tarafından gönderilir. Ancak alıcı, bir kişi ya da grup olabilir. Bu sebeple Mesaj.alici ikincil anahtarı hem Grup.id birincil anahtarı, hem de Kullanici.id birincil anahtarıyla ilişkili gözükmektedir. Fark ettiyseniz bu iki ilişkide birincil anahtara yapılan bağlantıda zorunluluk bulunmamaktadır. Bunun sebebi ikincil anahtarın, birincil anahtarlardan birine gidecek olmasıdır. Dolayısıyla aynı anda iki tarafa da bağlanmayacağı için iki bağlantının da oluşmama olasılığı vardır.

Peki sistem kurgusunda bu ikincil anahtarın Grup ya da Kullanici varlığına bağlanacağını nasıl anlayabiliriz? Mesaj.tur niteliğinin amacı bunu belirlemektir. Bu nitelik kişisel ya da benzeri bir değer aldığında bağlantının Kullanici varlığıyla yapılması gerektiğini; grup değerini aldığında ise Grup varlığıyla yapılması gerektiğini gösterebilir. Bu durum, pek de sık karşılaşılan bir durum olmamakla birlikte pek deneyimli olmayan tasarımcıların genellikle birden fazla Mesaj varlığıyla çözmeye çalıştığı bir problemdir. Ancak önerimizde gördüğünüz üzere tek bir Mesaj varlığı tüm sistem mesajlarını yönetmek için yeterlidir. Ayrıca nesne tabanlı yaklaşım açısından bakıldığında da doğrusu budur.

Sistem önerisinde bir mesajın okunma bilgisini tüm kullanıcılar için tuttuğumuz bir yapı önerdik. Peki sadece alıcının okuma bilgisini kaydetmek isteseydik?

Okundu varlığını tamamen ortadan kaldırarak, özel mesajlaşma durumunda alıcının mesajı okuyup okumadığını kayıt altına alacak; grup mesajlaşması için herhangi bir okundu bilgisi tutmayacak bir değişiklik önerisinde bulunun.

Bu seviyeye gelmiş bir okuyucunun bu problemi rahatlıkla çözeceğinden emin olduğum için son atölyenin yanıtını bulmayı sizlere bırakıyorum.

8.7. Ses Tabanlı Sosyal Medya Yazılımı

8.7.1. Problem

Bir girişim (start-up) ekibinde yer alıyorsunuz. Girişimin konusu yeni bir sosyal medya platformu. Bu platformda tüm kullanıcı gönderileri birer ses kaydı olmalı. Bu yeni girişimde veri tabanı tasarımcısı olarak yer almaktasınız. Geliştirilecek sosyal medya platformuna ait ayrıntılar aşağıda verilmiştir.

1) Platform kullanıcıları ses kaydı içeren iletiler paylaşabilirler. İletiler metin içerik alamazlar. Yalnızca ses kaydı içermelidirler.

2) Gönderiler altına metin ya da ses içeren yorumlar yapılabilir.

3) Gönderiler için beğenme ya da beğenmeme işareti koyulabilir.

Belirtilen özelliklere sahip bir platform geliştirmek için gerekli veri tabanı tasarımını gerçekleştirin.

8.7.2. Çözüm Önerisi

Popülerliği ve sayıca çokluğu sebebiyle böyle bir girişim ve bunun için hazırlanacak yazılım projesinde yer alma şansınız görece daha fazladır. Girişim projeleri için yazılım geliştirmek konusundaki en büyük risk, genellikle daha önce yapılmamış işler denendiği için olası aksiliklerin öngörülmesinin zor olmasıdır. Önünüzde daha önce aynı amaçla geliştirilmiş bir proje olduğunda bu projeyi örnek alarak geliştirme yapmak süreci hızlandıracaktır.

Ancak tüm kuralları siz koyuyorsanız ve bazı yanıtlanması gereken sorular henüz yanıta sahip değilse geliştirme sürecinde öngörülemez zorluklar yaşanması muhtemeldir. Ele aldığımız girişimcilik örneği için kendi yöntemimizi kullanarak kelime analizimize başlayalım. Problem başlığı altında gördüğümüz ad özelliğindeki kelimelerden seçtiklerimiz aşağıda verilmiştir.

Kullanıcı, ses kaydı, ileti, metin, ses

İşlevsel özellikleri belirlemek adına seçtiğimiz fiil özellikli kelimeler de şu şekilde seçilmiştir:

Yorum yapmak, beğenmek, beğenmemek

Sistem içerisinde kullanıcıların kaydının tutulması gerekliliği kolayca kabul görecektir diye düşünüyorum. Bu sebeple ilk kelimemiz olan Kullanici, ilk varlığımız olarak değerlendirilebilir. Ses kaydı ifadesi, bu platformda gönderilecek iletilerin türü olarak karşımıza çıkmaktadır. Bu sebeple Ileti adlı bir varlık tanımlamalı ve bu iletinin bir ses kaydı içerdiğini unutmamalıyız. Böylece Ileti varlığını şu şekilde tanımlayabiliriz:

Ileti: id, dosyaAdi, timestamp

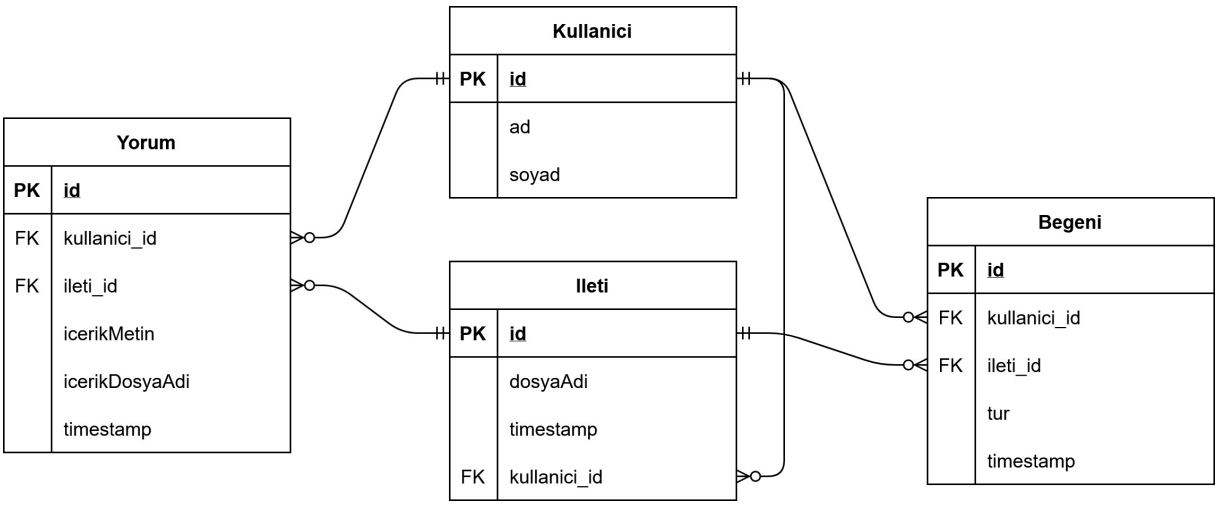
Ses kaydı olarak bahsedilen içerik, bir multimedya içeriğidir ve ayrıca bir dosya olarak saklanmaktadır. Bu tarz dosyaların veri tabanı içerisinde saklanması imkansız olmasa da pratikte neredeyse hiç tercih edilmeyen bir yöntemdir. Metin içeriğe göre devasa sayılabilecek boyuttaki dosyaları olduğu gibi bir veri tabanı içerisine yerleştirmek, bir süre sonra veri tabanını yönetmeyi oldukça zorlaştıracaktır. Hantallaşma, beraberinde yedekleme, işleme ve diğer bazı sorunları da getirebilecektir. Bu sebeple multimedya dosyaları genellikle bir sunucu üzerine kaydedilir ve bu dosyaya erişim yolu veri tabanı içerisinde saklanır. Ileti.dosyaAdi niteliği, bu ileti için oluşturulmuş olan ses kaydı dosyasının sunucu üzerindeki yolunu gösteren metinsel bir ifadedir. Böylece ses kaydı dolaylı olarak veri tabanında saklanmış olur ancak dosyanın kendisini taşımanın getireceği yükten uzak kalmak mümkün hale gelir.

Metin ve ses ifadeleri, yorum için kullanılan kelimelerdi. Dolayısıyla onları yorumlarla ilişkilendirmek üzere bekletelim bir süre. Fiillere geçtiğimizde ilk kelimemiz yorum. İletilere yapılacak yorumların mutlaka bir varlık içerisinde tutulması gerekiyor. Bu işlevsel bir tablo olduğu için hangi ilişkiden doğacağını da ele almak gerekir. Bir yorum, bir ileti için bir kullanıcı tarafından yapılmalıdır. Dolayısıyla Yorum varlığı, Kullanici ve Ileti varlıklarının çoğa çok bağlantısının sonucunda oluşacak bir bağlantı varlığı olarak da görülebilir. Az önce beklemeye aldığımız nitelikleri de işin içine katarsak Yorum varlığımızı şu şekilde kurgulayabiliriz:

Yorum: id, icerikMetin, icerikDosyaAdi, timestamp

Yorum varlığını bu şekilde kurgulamak, yorumların hem metin içerikli hem de ses kaydı eklenebilir olmasını sağlayacaktır. icerikMetin niteliği metin içeren yanıtı, icerikDosyaAdi ise ilgili ses kaydı dosyasının sunucu üzerindeki yolunu saklayacak olan niteliktir.

Son olarak beğenme ve beğenmeme durumlarını ele alalım. Begeni adlı tek bir varlıkla hem beğenme hem de beğenmeme durumlarını kayıt altında tutabiliriz. Bu varlık da Yorum varlığına benzer şekilde Kullanici ve Ileti varlıklarının çoğa çok bağlantısının bağlantı varlığı olarak görülebilir. Begeni varlığına ekleyeceğimiz bir tur niteliği, etkileşimin beğenmek ya da beğenmemek olduğu bilgisini tutabilir. Tüm bu değerlendirmeler neticesinde oluşturduğumuz veri tabanı tasarımımız Şekil 8.7'de sunulmuştur.



Burada Kullanici ve Ileti adlı iki temel varlıktan söz edebiliriz. Yorum ve Begeni varlıkları, Kullanici ve Ileti varlıklarının çoğa çok bağlantılı olması neticesinde ortaya çıkan bağlantı varlıkları olarak da değerlendirilebilirler. Daha önce de değindiğimiz üzere, iki varlık birden fazla kez çoğa çok bağlantılı olabilir. Çoğa çok bağlantı genellikle bu iki varlık arasındaki bir işlevi konu almaktadır. Burada da kullanıcılar ile iletilerin iki farklı etkileşim durumu iki bağlantı varlığıyla kayıt altına alınmıştır.

Burada bir ayrıntı üzerinde duralım. Eğer iki varlık arasındaki ilişkiyi arıyor ve çoğa çok ilişki ile karşılaştıysanız, bu iki varlığın hangi işlev ile bu ilişkiyi kurduğu üzerinde biraz durun. Sistemde yer alması gereken ancak öngöremediğiniz bir varlık, karşılaştığınız ilişki için oluşturacağınız bağlantı varlığı olabilir.

8.8. Video Paylaşım Platformu

8.8.1. Problem

Bir video içerik yayınlama platformunun geliştirme ekibinde yer alıyorsunuz. Yönetim tasarından sistemin yenilenmesine karar verildi ve yeniden yazılacak bu sistem için veri tabanı tasarımcısı olarak ekipte yer almaktasınız. Kullanıcıların ücretsiz ve sınırsız şekilde video paylaşmalarına olanak sağlayan bu sistemin mevcut çalışma yapısına ait ayrıntılarıyla birlikte geliştirilecek yeni özelliklere ait bilgiler birleştirilerek liste olarak, aşağıda verildiği şekliyle size sunulmuştur.

1) Kullanıcılar bir ya da daha fazla içerik kanalı oluşturabilir. Kanallar, kişilerin paylaştığı videoları içeren özel sayfalardır.

2) Kullanıcılar kanalları takibe alabilirler.

3) Yalnızca kanal sahibi kullanıcı, kanala içerik yükleyebilir.

4) Tüm içerik izlemeler kayıt altına alınmalıdır.

5) Kullanıcılar videolara beğenme ya da beğenmeme işareti koyabilirler.

6) Videoların toplam izlenme sayıları kayıt altında olmalıdır.

7) Tüm video içerikler bir video dosyasına, başlığa ve açıklamaya sahiptirler.

Verilen ayrıntılar ışığında bir veri tabanı tasarımı hazırlayınız.

8.8.2. Çözüm Önerisi

Talep edilen veri tabanı tasarımı için problem başlığı altındaki ayrıntıları inceleyerek kelime analizimizi gerçekleştirdiğimizde elde ettiğimiz kelimeler, aşağıda listelediğimiz şekildedir. Önceki örneklerden farklı olarak burada fiil özelliğine sahip yani sistem için bir işlev belirten kelimeleri ayrıca listelemeyip, aşağıdaki liste içerisinde kelime başında bir * işareti kullanarak vurguladık.

Kullanıcı, içerik, kanal, video, *paylaşmak, *takip, sahip,

*içerik yükleme, *izleme, *beğenme, *beğenmeme, izleme sayısı, dosya, başlık, açıklama

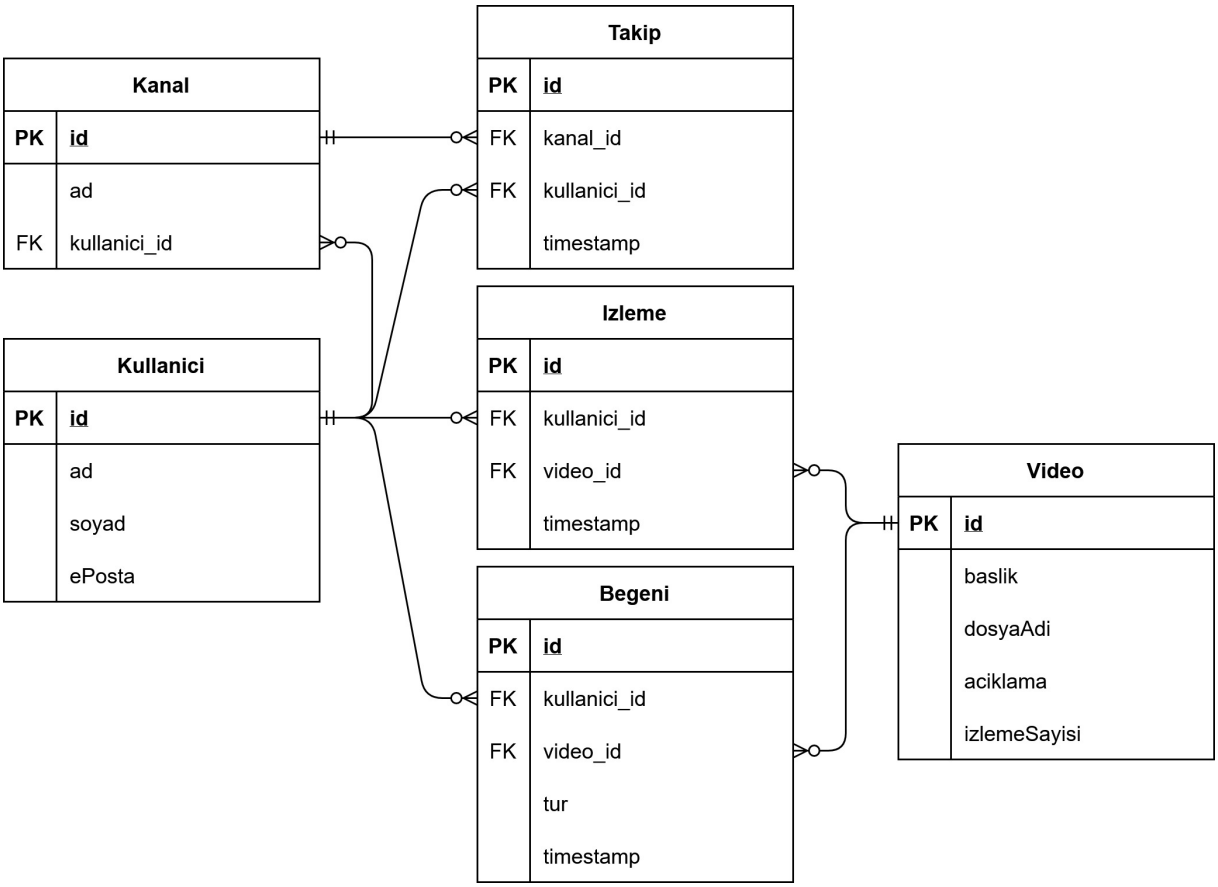
Burada kullanıcı ve sahip kelimelerini tek bir Kullanici varlığı tanımlayarak ifade edebileceğimiz konusunda kolayca hemfikir olacağımızı düşünüyorum. İçerik kelimesiyle kastedilen video içerikler olduğu için bu iki kelimeyi birlikte ele alarak Video varlığının tanımlanabileceğini de söyleyebiliriz. Bu videoların bazı kanallara yükleneceğinden bahsedilmektedir. Kolayca bunu da bir Kanal varlığı oluşturarak ele alabiliriz. İzleme sayısı, dosya, başlık ve açıklama kelimelerinin paylaşılan video içeriğe ait nitelikler olduğunu kabul edebiliriz. Böylece sistemimizde yer alan olgulara ait varlıkları tanımlamış olmaktayız.

İşlev anlamı taşıyan kelimeleri ele aldığımızda bazı kelimelerin yazılım katmanında ele alınması gereken kelimeler olduğunu görmekteyiz. Örnek olarak paylaşmak ve içerik yükleme ifadeleri yazılım ve dolayısıyla arayüz katmanında ele alınması gereken sistem özelliklerindendir. Bunlarla ilgili veri tabanında gerçekleştireceğimiz herhangi bir kayıt işlemi bulunmamaktadır. Takip kelimesi kullanıcıların kanalları takip etmesi özelliği sebebiyle kullanılmıştı. Bu kelime gerçekten de kullanıcının gerçekleştireceği takip işlevini temsil etmekle birlikte Kullanici ve Kanal varlıkları arasındaki çoğa çok bağlantı için kullanılabilecek bir bağlantı varlığı olma potansiyeli de taşımaktadır.

İzleme işlevi, bir kullanıcı ile bir video etkileşimiyle oluşmaktadır. Kullanıcıların videoları izlemesi ya da videoların kullanıcılar tarafından izlenmesi olarak düşünüldüğünde burada da çoğa çok bağlantı ve bu bağlantıyı sağlayan bir bağlantı varlığından bahsedilebilir. Elbette burada bağlantı varlığı İzleme olarak tanımlanmalıdır.

Beğenme ve beğenmeme durumları iki benzer işlevin türünün farklı olması olarak düşünülebilir. Beğenme ya da beğenmeme durumu aynı zamanda bir içeriğe pozitif ya da negatif puan vermekle teknik olarak aynı durumdur. Bu sebeple iki işlevi ayrı varlıklar yapmak yerine tek bir Begeni varlığı olarak ele alabiliriz. Bu varlık kullanıcılar ile videoların bir başka etkileşimi olarak görülebilir.

Tüm bu yorumlar neticesinde elde edeceğimiz veri tabanı tasarımı Şekil 8.8 ile sunulmuştur.



Burada Kullanici ve Kanal varlıkları arasındaki sahiplik bağlantısı dışında tüm bağlantıların çoğa çok olduğunu dikkatinize sunarım. Kullanici ve Video varlıklarının iki kez, Kullanici ve Kanal varlığının bir kez

çoğa çok bağlantı kurması neticesinde 3 tane bağlantı, dolayısıyla işlev varlığı tanımlanmıştır. Sistem işlevlerini içeren bu varlıkların, olgu varlıkları olmasa da sistemin ana işlevlerini üstlenen, çok önemli varlıklar olduklarını görmekteyiz. Bu da çoğa çok bağlantı için mecburen tanımlanan bağlantı varlıklarının aslında önemli rollere sahip; sistemin işleyişi açısından önem arz eden varlıklar olduklarını göstermektedir.

8.9. Bulut Tabanlı Dosya Barındırma Yazılımı

8.9.1. Problem

Kullanıcıların dosyaları arşivleyebilmesini sağlayacak bir bulut hizmetin geliştirilmesinde ekip üyesi olduğunuzu farzedin. Sizden, aşağıda özellikleri verilmiş olan sistem için bir veri tabanı tasarımı gerçekleştirmeniz beklenmektedir.

1) Sisteme kaydolan kullanıcılar, farklı kapasite seçenekleri içeren paketlerden birini satın almak zorundadır.

2) Her kullanıcı, satın aldığı paket doğrultusunda bir kapasite kullanma hakkına sahiptir.

3) Kullanıcılar, kendi arşivleri içerisinde istedikleri sayıda ve kırılımda klasör açabilir, istedikleri türde dosyalar yerleştirebilir.

4) Kullanıcı tarafından silinen dosyalar doğrudan silinmez, çöp kutusuna taşınırlar.

5) Çöp kutusuna taşınan dosyalar 30 gün sonra geri çekilmedikleri takdirde tamamen silinirler.

6) Dosyalar paylaşıma açılabilirler. Paylaşım ve paylaşılan dosyalara erişim, e-posta adresleri kullanılarak gerçekleştirilir.

Bu şartlara uyan bir veri tabanı tasarımı gerçekleştiriniz.

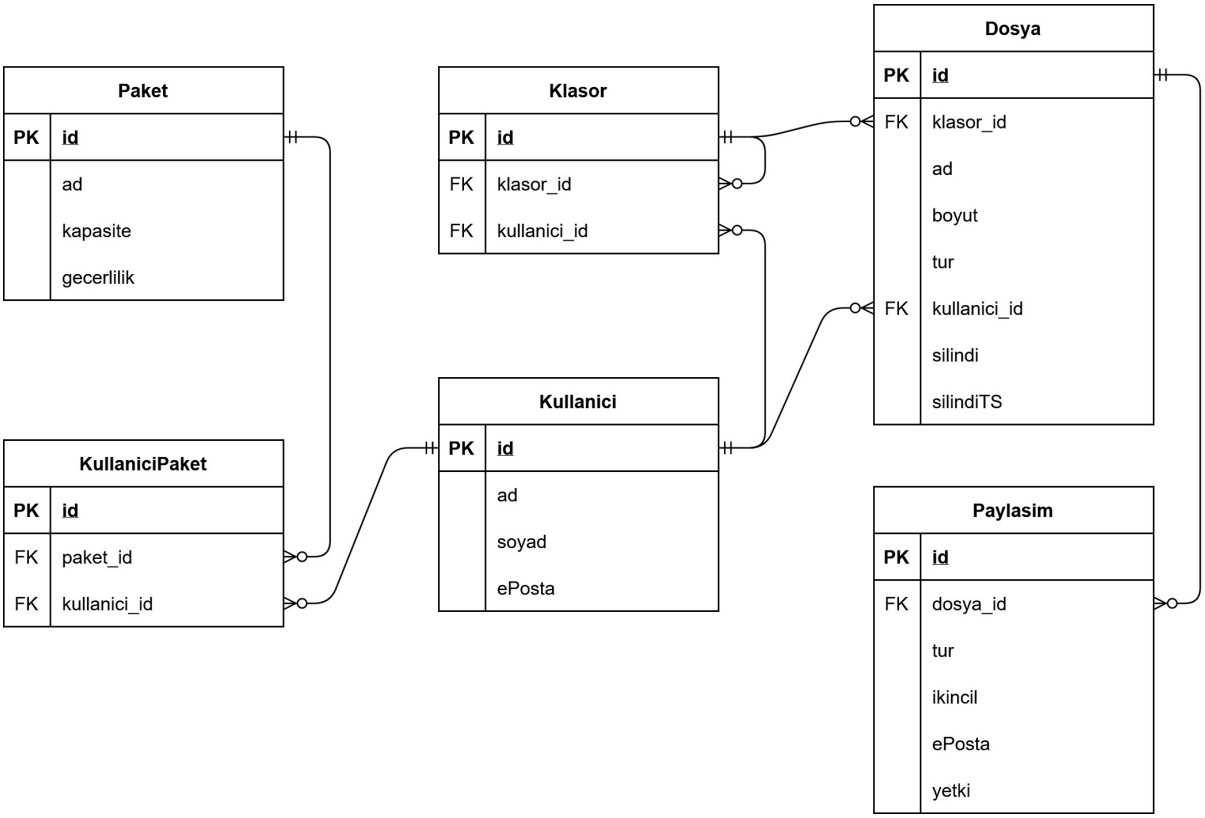
8.9.2. Çözüm Önerisi

Ele alınan problem için kelime analizini gerçekleştirdiğimizde aşağıdaki kelimeleri elde etmekteyiz. İşlev niteliğinde olabilecek kelimeler, kelimenin başına eklenen bir * işareti ile vurgulanmıştır.

Kullanıcı, dosya, arşiv, kapasite, paket, *satın almak, klasör, tür, *silmek, çöp kutusu, *taşımak, *geri döndürmek, *paylaşmak, e-posta

Sistem kullanıcılarının kaydedilmesi için Kullanici adlı varlık, tanımlayacağımız ilk varlıktır. Dosya ve klasörlerle ilgili yapı daha önce özel durumlar altında örneği verilmiş bir yapı idi. Klasör yapılandırması sonsuz sayıda kırılıma sahip olabilecek bir hiyerarşi sunduğu için farklı seviyedeki kategorilerin farklı varlıklar ile sunulması etkili bir çözüm değildir. Bu sebeple klasörlerin kayıt altına alınacağı varlık kendi kendisiyle bire çok bağlantı kurarak sınırsız sayıda alt kırılıma sahip olabilir. Bu yapı Klasor adlı varlık ile kayıt altına alınacaktır. Dosyalar ise klasörlere bağlı olarak kayıt altına alınması gerektiği için Dosya adlı bir varlık ile kayıt altına alınacaktır. Arşiv kelimesi yine sistemin tamamını niteleyen, bir kullanıcının klasör ve dosyaları bütününe verilen bir ad olduğu için varlık olarak tanımlama ihtiyacı oluşturmayan bir kelimedir. Kapasite ve paket kelimeleri, kullanıcıların bu sistemde faydalanabilme limitlerini göstermektedir. Bir Paket varlığı kullanarak kullanılabilecek kapasite türleri gruplandırılabilir. Kapasite, Paket varlığının bir niteliği olmalıdır. Ayrıca kullanıcılar zaman içerisinde farklı paketlere abone olabileceği için Paket ile Kullanici varlıkları arasında çoğa çok bağlantı olduğunu belirtmek gerekir. Bu ilişkideki bağlantı varlığı KullaniciPaket olarak tanımlanabileceği gibi işlevsel bir ad olarak Abonelik de tercih edilebilir. Aynı zamanda kelime listemizdeki satın alma işlevi de bu bağlantı varlığı ile karşılanabilmektedir. Tür ifadesi bir olgunun özelliği olduğu için ilgili varlık altına nitelik olarak kolaylıkla eklenebilir. Bir dosyanın silinip silinmeme bilgisi eklenecek bir nitelik ile kayıt altına alınabilir. silindi adlı bir niteliğin 1 değeri alması neticesinde ilgili dosyanın silinip, çöp kutusuna gönderildiği düşünülebilir. Böylece bir dosyayı çöp kutusuna göndermek için yapılması gereken sadece Dosya.silindi = 1 tanımlaması yapmaktır. Ayrıca bir süre geçtikten sonra çöp kutusundaki dosyanın tamamen silinmesi için dosyanın çöp kutusuna ne zaman gönderildiğinin de

bilinmesi gerekmektedir. Bunun için ise silindiTS adlı bir nitelik yardımıyla dosyanın silinme zamanına ait timestamp değeri kayıt altına alınabilir. Böylece yazılım katmanında, tanımlı kurtarma süresi dolan dosyalar otomatik olarak sistemden tamamen silinebilirler. Dosyaların paylaşılabilmesi, doğal olarak dosyaların bir niteliği olarak görülebilir. Ancak bir dosyanın birden fazla kez paylaşılabilmesi mümkün olabileceği için paylaşım işlevinin Dosya varlığı ile bire çok ilişkili olması gerekmektedir. Paylaşım bilgisinin kayıt altına alınabilmesi için Paylasim adlı bir varlık tanımlamamız uygun olacaktır. Bu ayrıntılı açıklamalar neticesinde önerdiğimiz veri tabanı tasarımı Şekil 8.9 ile sunulmuştur.



Tasarım, üç önemli bileşenden oluşmaktadır. Birincisi Klasor-Dosya yapılanmasının kayıt altına alınmasıdır. Bu özel bir durumdur ve dinamik bir kayıt gerçekleştirilebilmesi için kullanılması gereken yapı şekildeki gibidir. Özellikle Klasor varlığının kendi kendisiyle yaptığı bire çok ilişkiyi inceleyiniz. İkinci önemli bileşen üyeliklerin paketlere abone olmasıdır. Abonelik bilgisi KullaniciPaket varlığı ile kayıt altına alınmaktadır.

Üçüncü ve son bileşen ise dosyaların paylaşıma açılabilmesidir. Bu özellik Paylasim varlığı ile kullanılabilmektedir. Paylasim varlığının adı ve yapısı gereği bir bağlantı varlığına benzediği dikkatinizi çekmiştir. Aslında bu yapıda da bağlantı varlığı olarak değerlendirilebilir. Dosyaların, paylaşılan kişiler ile ilişkisini sağlayan bir işleve sahiptir. Ancak dosyaların paylaşıldığı kişiler bu sistemde kayıt altında olmadığı; paylaşımın e-posta adresleri ile sağlandığı için Paylasim varlığının paylaşılan kişiyle ilgili bir varlık ilişkisi bulunmamaktadır. Bunun yerine Paylasim.ePosta niteliğinin mantıksal olarak bir ikincil anahtar özelliği taşıdığını söyleyebiliriz. Eğer paylaşılan kişilerin de sisteme kayıtlı olma zorunluluğundan bahsetseydik bu kişileri de Kullanici varlığı içerisine kaydedecektik ve Paylasim varlığı Dosya-Kullanici varlıklarının çoğa çok ilişkisinin bağlantı varlığı haline gelecekti.

8.10. Satranç Turnuvaları Yazılımı

8.10.1 Problem

Ülkenizin ulusal satranç fedarasyonunda bilişim bölümünde görev yapmaktasınız. Yönetim, tüm satranç turnuvalarının, bu maçlarda oynanan oyunların ve lisanslı oyuncuların kayıtlarını tutabilecekleri bir bilgi sistemi geliştirmek istiyor. Sistem, aşağıda listelenmiş özelliklere sahip olmalı. Sizden, veri tabanı tasarımı gerçekleştirme konusunda görev almanız isteniyor.

1) Satranç maçları iki taraftan birinin kazandığı ya da berabere kalındığı maçlardır. Kazanç durumu 1 puan, beraberlik 1/2, kayıp 0 puan getirmektedir.

2) Her bir oyuncu elo adı verilen bir oyuncu skoruna bağlıdır. Bu skor, oyuncunun güncel durumunu gösterir ve maç sonuçlarına göre güncellenir.

3) Maçlar puanlı ya da puansız olabilir. Yalnızca puanlı maçlar oyuncuların elo skorunun değişmesine sebep olur.

4) Oyuncuların bir turnuvaya ön kayıt yaptığı bilgisi kayıt altında olmalıdır.

5) Oyuncuların maç sonuçları turnuvaya bağlı olarak kayıt altında olmalıdır.

Bu şartlara uygun veri tabanı tasarımını gerçekleştiriniz.

8.10.2 Çözüm Önerisi

Varlıkların ve sistem işlevlerinin belirlenebilmesi için problem başlığında verilen ayrıntıları analiz ettiğimizde listelenen kelimeleri elde edebiliriz:

Turnuva, *maç, oyuncu, puan, elo, *güncellemek, puanlı,

puansız, *ön kayıt, maç sonuçları

Problem tarifi içerisinde turnuvalar düzenleneceği ve bu turnuvalar kapsamında maçlar gerçekleştirileceği açıkça ifade edilmektedir. Bu sebeple Turnuva ve Mac varlıklarını tanımlamamız gerektiğinden bahsedebiliriz.

Oyuncuları ve gerekli olması durumunda sisteme dahil olabilecek diğer kişileri de kaydetmek üzere Kisi adlı bir varlık tanımlayabiliriz. Burada puan ve diğer adıyla elo ifadesi oyuncunun bir özelliği olduğu için Kisi varlığı içerisinde nitelik olarak tanımlanabilir.

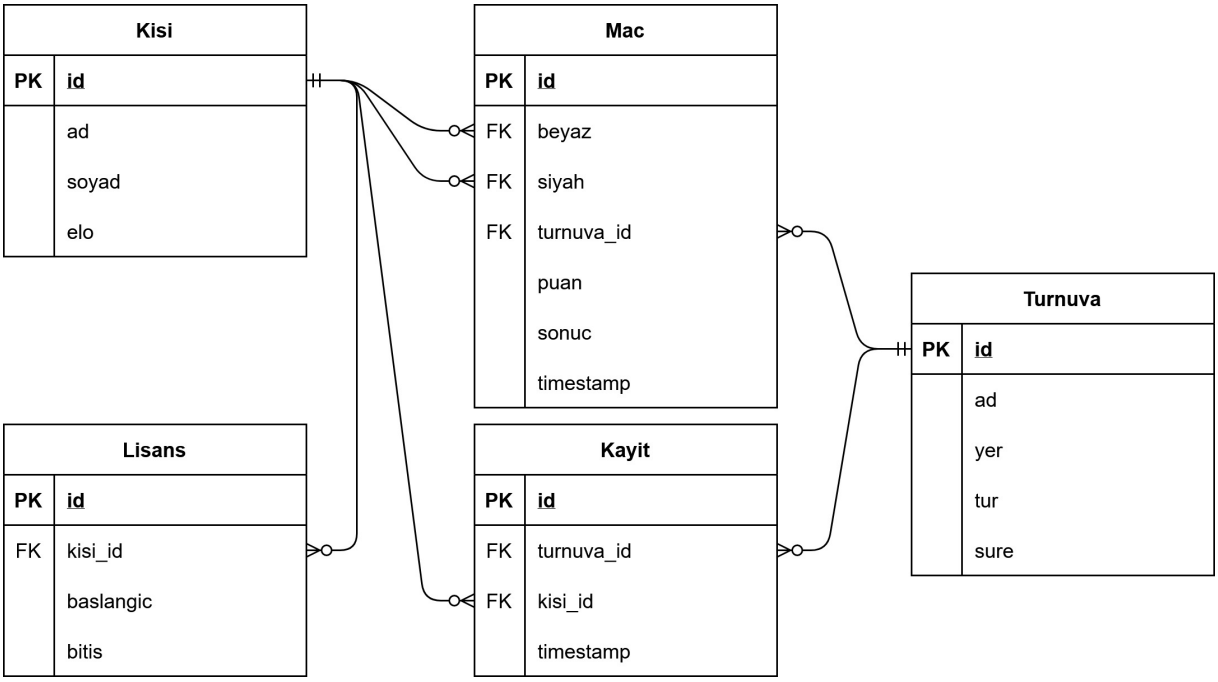
Güncellemek ifadesi, oyuncuların maç sonuçlarına göre puanlarının güncellenmesi anlamına gelmektedir. Bu işlev, yazılım katmanında gerçekleştirilmesi gereken bir işlem olduğu için ve sonucu Kisi.elo niteliği içine kaydedilebileceği için ek bir müdahale gerekmemektedir.

Puanlı ve puansız ifadeleri gerçekleştirilen maçın özelliği olmakla birlikte aslında turnuva şartları ile tanımlanan bir kuraldır. Dolayısıyla bu durum Turnuva varlığı içerisinde tur adlı bir nitelik ile kayıt altına alınabilir. Bu bilginin Mac varlığı altında tutulması veri bütünlüğü açısından bir fark yaratmazdı. Ancak bir turnuvaya bağlı tüm maç kayıtlarında tur niteliğinin aynı değeri alması beklenirdi. Bu da aynı değerin maç sayısı kadar tekrar etmesi anlamına gelecektir. İlgili kaydın Turnuva varlığı altında tutulması bu veri tekrarı problemini ortadan kaldıracaktır.

Ön kayıt işlemleri, Turnuva varlığı ile Kisi varlığı arasındaki çoğa çok ilişki sonucunda oluşan bir bağlantı varlığı olarak ele alınabilir. Bir turnuvaya birçok kişi ön kayıt yaptırabileceği gibi bir kişi birçok farklı turnuvaya kaydolabilir. Bu durumda kayıtları içeren varlık, ilişkili olduğu bu iki varlığa bağlı bir varlık olarak karşımıza çıkacaktır.

Maç sonuçları, Mac varlığıyla bire bir ilişkili olduğu için ayrıca bir varlık olarak tutulmasına gerek bulunmamaktadır. Burada Mac ve MacSonuc gibi bire bir ilişkili iki varlık tanımlamanın da hata olmayacağını belirterek önerilen çözümde bu iki varlığın tek bir varlık olarak ele alındığını belirtelim.

Değerlendirmeler neticesinde önerilen veri tabanı tasarımı Şekil ile sunulmuştur.



Bölüm Özeti

Bölüm içerisinde 10 özel probleme yönelik 10 farklı veri tabanı tasarımı gerçekleştirdik. Bunları yaparken aşağıdaki yaklaşımlar bizim için çok önemliydi.

1. İhtiyacı dinlemek: Sistem analizi ve tasarımı aşamasında bu sistemi kullanacak kişileri dikkatle dinlemek ve gerekli soruları belirleyip sormak çok önemlidir. Bu yazılım yokken süreci nasıl işlettiklerini öğrenmek en faydalı yöntemlerden biridir.
2. Kelime analizi yapmak: Aldığınız notlar üzerinde kelimeleri ayıklamak, bunları ad ve fiil olarak ikiye ayırmak iyi bir başlangıç noktası elde etmenizi sağlayacaktır. Adlar tablo adlarını, fiiller ise bağlantı tablolarının adlarını belirlememizde yol göstericidir.
3. İlişki tiplerini doğru belirlemek, veri bütünlüğünü sağlamak açısından oldukça önemlidir.

Bölüm içerisinde aşağıdaki konularda veri tabanı örnekleri sunulmuştur:

1. Diyetisyenler için platform
2. Araç satış portalı
3. Para transferi yazılımı
4. Nakliye firması organizasyon yazılımı
5. Operatör Koordinasyon yazılımı
6. Elektronik iletişim yazılımı
7. Ses tabanlı sosyal medya yazılımı
8. Video paylaşım platformu
9. Bulut tabanlı dosya barındırma yazılımı
10. Satranç turnuvaları yazılımı

Tüm bu örnekler için önce kendiniz bir tasarım üretmeyi deneyip, sonrasında yorumları incelediyseniz bu konuda büyük gelişme kaydettiğinize emin olabiliriz. Artık bilgisayar programcılığında bir adım daha öndesiniz.

Kaynakça

Akadal, E. 2020. Veritabanı Tasarlama Atölyesi. Türkmen Kitabevi, İstanbul.

ARKA KAPAK YAZISI

Veri tabanı tasarımcılığı, bilgi, beceri ve deneyimin bir karması ile elde edilebilen bir vasıftır. Bu ders içerisinde, veriyi tanımak ve verinin enformasyona dönüşümünden başlayarak veriyi en uygun şekilde elektronik ortamda saklayabilmek için dikkat edilmesi gereken tüm ayrıntılar birer birer ele alınmıştır. Veri tabanı tasarımcılığının ancak deneyimle öğrenilebileceği görüşü paralelinde her bölüm içerisinde yaparak öğrenme yöntemiyle anlatım sağlanmıştır.

Dersi başarıyla tamamlayan bir öğrenci, bir yazılım geliştirme projesinde veri tabanı tasarımcısı olarak görev yapabilecek yetkinliğe sahip olabilecektir. Bu dersten en fazla faydayı sağlamanın yolu, her bir problemi bireysel olarak da inceleyerek çözüm üretmek; yalnızca okuyarak tamamlamamaktır.

Dersin son ünitesi gerçek hayata yönelik çok sayıda örnek ve bu örnekler için veri tabanı tasarımları içermektedir. Ayrıca bu bölümle birlikte bir problemle karşılaşıldığında nasıl ele alınması gerektiğine dair yol gösterici ipuçları bulunmaktadır.

Bu dersin, geleceğin bilgisayar programcılarına faydalı olması dileğiyle.