3. GÖSTERİM ŞEKİLLERİ VE TERİMLER

Birlikte Düşünelim

Bilgisayar programlama sürecinin tamamı bilgisayar ortamında mı gerçekleştirilir?

Veri tabanı tasarımı dersi için “tasarım” kelimesi özellikle seçilmiştir. Tasarım kelimesini kullanınca ne hayal ediyorsunuz? Tasarım süreci doğası gereği hangi faaliyetleri barındırır?

Bir veri tabanı tasarımı gerçekleştirmekle sanat eseri tasarlamak arasında nasıl benzerlikler bulabiliriz?

Veri tabanı tasarımında “bilgelik” yeterli midir?

Bir veri tabanının performansı nasıl ölçülebilir?

Başlamadan Önce

Yazılım geliştirme -eğer hobi projesi değilse- bir ekip işidir. Aynı zamanda genellikle kısa bir süreç de değildir. Yazılım geliştirme süreci boyunca ekibin hangi amaç ve motivasyonla yazılım geliştirdiği ve önceliklerin neler olduğunu; yol haritasının ne şekilde tanımlandığının bilinmesi oldukça önemlidir. Bunun için de yazılım projelerinde sistem analizi ve tasarımı ile buna bağlı olarak dokümantasyon süreçleri oldukça önemlidir. Bir yazılımcı için dokümantasyon oluşturma ve inceleme zaman zaman sıkıcı olabilse de projenin sağlıklı yürütülmesi açısından gereklidir.

Veri tabanı tasarımı da sistem analizi ve tasarımı sürecinin bir parçasıdır. Bu sebeple henüz bilgisayar ortamında geliştirmeye başlanmadan önce yapılan çeşitli faaliyetler içerir. Bu faaliyetler neticesinde de bir dokümantasyon elde edilir. Elde edilen çıktının tüm proje ekibi ve potansiyel diğer ekip üyeleri tarafından rahatlıkla anlaşılabilir ve uygulanabilir olması gerekmektedir. Bunu sağlamak için ortak kullanılan çeşitli kavram ve diyagramlar bulunmaktadır. Bu bölüm içerisinde temel kavramların anlamları ve gösterim şekilleriyle ilgili temel bilgileri edinebilecek; bu tür teknik dokümanları oluşturmak ve incelemek için gerekli altyapıyı edinmiş olacaksınız.

3.1. Terimler

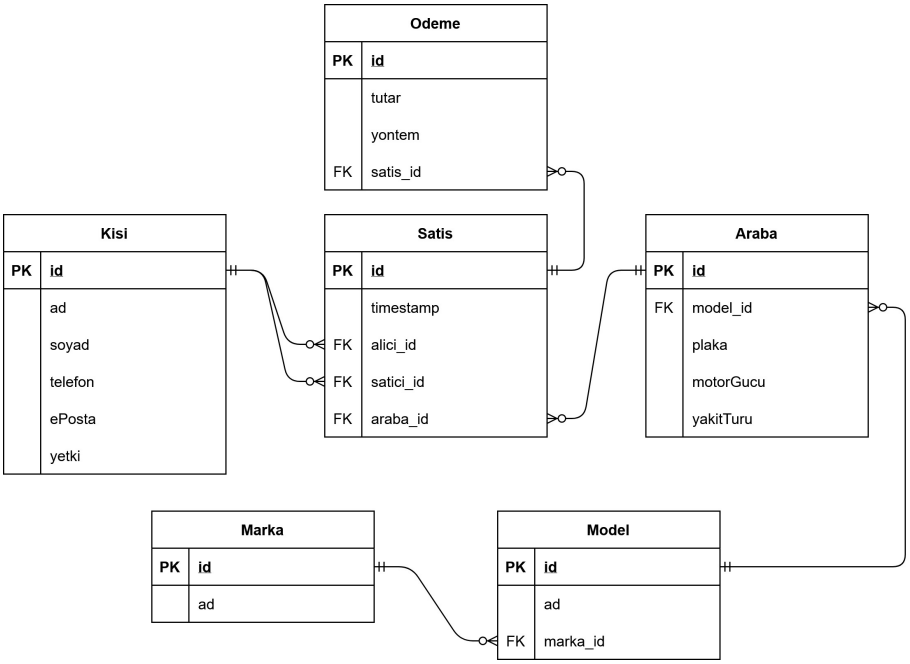
Bir bilişim sistemini hayata geçirme süreci baştan aşağıya sinerji içermektedir. Her bir mikro sürecin çıktısı diğer süreçlere girdi olmaktadır. Tüm süreçler birbirlerini beslerler ve süreçler tamamlandığında ihtiyaç olunan sistem hayata geçmiş olur. İhtiyaç duyulan sistemin geliştirilmeye başlanılmasından önceki geniş hazırlık sürecini sistem analiz ve tasarım süreci olarak ele alıyoruz. Veri tabanı tasarlama süreci de sistem analizi ve tasarımı sürecine dahil bir mikro süreç olarak karşımıza çıkıyor. Bilişim teknolojilerin deyince tüm sürecin bilgisayar ortamında gerçekleştirildiği düşünülüyor olabilir. Ancak yazılımın geliştirme adımına gelene kadar gerçekleştirilen faaliyetler, temelde bir tasarım sürecinin parçaları olduğu için bu süreç daha çok iletişime ve dokümantasyona dayalıdır. Bu aşamada süreç teknik anlamda bilgisayarda uzakta gerçekleştirilir. Elbette iletişim ve dokümantasyon için de bilgisayar kullanılır ancak burada teknik anlamda diyerek vurguladığımız konu, programlama ve gelişmiş araçların kullanımıdır.

Tasarım sürecinin çıktısı, analiz ve tasarım süreci boyunca elde edilen bulguların belgelendirilmesidir. Bu belgeler, tüm geliştirme, test, iyileştirme süreçlerinde kullanılacak; sistemle ilgili gelecekte yapılacak iyileştirmeler için de yol gösterici olacaktır. Bu sebeple her ne kadar sistem analizi ve tasarımı yapan ekiple geliştirme yapan ekip doğrudan iletişimde olsa da elde edilen dokümante çıktılar evrensel geçerlilikte olmalıdır. Elde edilen bulgular, ekibin değişmesi durumunda, yeni ekip tarafından hızlıca anlaşılabilir nitelikte olmalıdır. Bu da aynı dili kullanmakla mümkündür. Bulguları dokümante edenler ile geliştiriciler aynı teknik terimleri kullanmalıdırlar. Bu sayede hazırlanan belgeler genel bir geçerliliğe sahip olur ve ekiplerdeki değişiklik belgelerin içeriğinin kalitesinde bir düşüşe sebep olmaz.

Bu dersin sonunda siz de bir veri tabanı tasarımcısı olacağınız için bu dokümanları oluşturabilme ve oluşturulmuş olanları gerektiği şekilde inceleyebilme becerisine sahip olmanız gerekmektedir. Bu başlık altında bir veri tabanı tasarım dokümantasyonu yapmak için gerekli terimler ve gösterim şekillerini ele alacağız.

Yeni öğrenilen bir olguyla ilgili terimlerin akılda tutulması biraz zorlayıcı olabilir. Bu sebeple her bir tanımı, şekli olarak da göstererek sunmak; öğrenme aşamasını kolaylaştıracaktır.

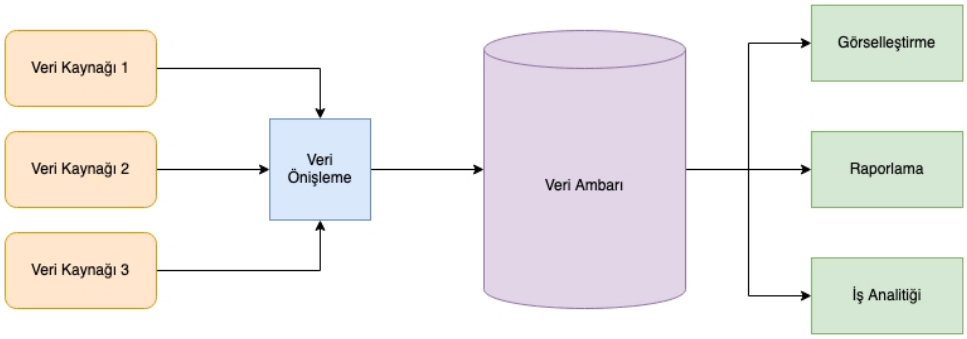
Dersin başından beri bahsediyor olsak da veri tabanı kavramıyla başlayalım. Veri tabanları, veri kümelerini yapılandırılmış biçimde, birbirleriyle ilgili sütunların gruplanarak farklı tablolara bölünmesi ve çeşitli ilişki türleriyle birbirlerine bağlanarak ihtiyaç halinde veriyi oluşturulan sorgular çerçevesinde hazırlayarak geri sunan yazılımsal altyapıdır. Bu tanım içerisinde yer alan olguları da teker teker inceleyeceğiz. Eğer bu tanım size yeterince anlamlı gelmediyse lütfen diğer tanımları inceledikten sonra tekrar bu tanımı inceleyiniz.



Veri kümesi ifadesi günlük hayatta veri seti (ya da veriseti) şeklinde de karşımıza çıkan, “dataset” kelimesinin Türkçe karşılığıdır. Bunu bir Microsoft Excel dosyası gibi hayal edebiliriz. Bir elektronik tablo içerisinde, ilk satırda sütunların hangi veriyi taşıdıklarını belirttikleri bir etiket, diğer satırlarda ise her bir satır ayrık bir kayıt olmak üzere, sütunlarda ilgili etikete denk gelen veriler bulunmaktadır. Veriyi saklamanın ve iletmenin geleneksel ve en basit yöntemlerinden biri veri kümesi kullanmaktır. En yüksek performansa sahip olmasa da kolaylık sağladığı için sık tercih edilen veri saklama ve aktarma türüdür.

Name	*Age*	*Role*
Sarah	19	Student
Janine	43	Professor
William	27	Associate Professor

Zaman zaman veri ambarı kavramıyla karşılaşmanız da mümkündür. “Data warehouse” teriminin Türkçe karşılığı olan veri ambarı, kısa tanım olarak merkezi bir veri tabanıdır. Biraz daha açıklamak gerekirse; birden fazla veri kaynağına erişimimiz varsa ve tüm veri kaynaklarını birleştirerek tek bir noktada enformasyon süreçlerini işletmek istiyorsak; tüm bu verilerin entegre olabilecekleri bir yapıda bir araya getirilmeleri gerekmektedir. Veri kaynaklarını bir araya getirdiğimiz yer veri ambarıdır.



Bir veri kümesini veri tabanına dönüştürmek için birbirleriyle ilgili sütunları gruplayarak birbirleriyle ilişkili birden fazla grup oluşturma sürecinden bahsetmiştik. Elde ettiğimiz her bir sütun grubuna tablo ya da varlık adı (table/entity) verilmektedir. Her bir veri tabanı tablosunu kendi içerisinde bir veri kümesi ya da elektronik tablo gibi değerlendirebilirsiniz. Tablolar basit anlamda satır ve sütunlardan oluşan ve veri içeren yapılardır. Veri tabanları, tablolara ek bazı özellikler katarlar. Bu da hem tabloların veri kümelerden farklılaşmasını hem de diğer tablolarla entegre olup veri bütünlüğünü sağlayabilmesine olanak sunarlar.

Kullanici	
PK	id
	ad
	soyad
	dogumYili

Şekilde Kullanici adında bir veri tabanı tablosunun gösterimini incelemektesiniz. Şekle göre şu bilgileri elde edebiliriz:

- Tablonun adı Kullanici'dir.
- Tablo 4 sütun içermektedir. Bunlar: id, ad, soyad, dogumYili
- Tabloda ayırt edici sütun id sütunudur. (PK işareti sayesinde bu yorumu yapabildik)

Veri tabanı tablosunun bu şekilde gösterilmesi durumunda içerisinde yer alan kayıtlarla ilgili bilgi sahibi olmak mümkün değildir. Ancak gerçekte, örnekteki gibi tarif edilen bir veri tabanı tablosunu göz önüne aldığınızda; id, ad, soyad ve dogumYili sütunlarından oluşan, içerisinde kayıtlar içeren bir veri tablosunu düşünebilirsiniz.

Şimdiye kadar "sütun" kavramı, elektronik tablolara aşina olmamız sebebiyle tercih edilmiştir. Ancak veri tabanları söz konusu olduğunda sütun olarak adlandırdığımız yapıya farklı adlar verildiğini de görmekteyiz. Bunlar sütun (column), nitelik (attribute) ve değişken (variable)'dir. Farklı kaynaklarda farklı şekillerde ele alınan bu terimlerin tümü, her bir tabloda yer alan dikey içeriği yani sütunu ifade etmektedir. Bu kitap ve ders kapsamında nitelik kelimesine öncelik verilecektir. Ancak yine de zaman zaman diğer karşılıkları da görmeniz mümkün. Bir ham veri kümesi basit bir şekilde sütunlardan ve bu sütunlara kayıtlar eklemek üzere satırlardan oluşmaktadır. Tüm verinin tek bir tablo şeklinde kaydedilmesi durumunda sütunlarla ilgili bir organizasyona ihtiyaç kalmaz. En fazla sıralamalarıyla ilgili değişiklikler yapılabilir ancak bu da verinin bütünlüğüne zarar verme konusunda bir risk barındırmaz. Ancak veri tabanı söz konusu olduğunda niteliklerin organizasyonu, veri tabanı tasarımının önemli bir parçası haline geliyor. Hangi niteliklerin birbirleriyle ilişkili olduğunu belirlemek, ilişkili nitelikleri gruplamak ve her bir grubu bir veri tabanı tablosu olarak tanımlamak sürecin önemli bileşenlerindendir.

Her bir tablo içerisinde yer alan satırı şimdiye kadar kayıt adıyla andık. Türkçe olarak kayıt adını kullanmaya devam edecek olsak da İngilizce dilindeki teknik karşılığının "tuple" olarak kullanıldığını bilmekte fayda var. Böylece her bir tablo/varlık içerisinde sütun/nitelik/değişkenlere karşılık kayıtlar bulunduğunu söyleyerek tüm terimleri bir arada tekrar kullanabiliriz.

Tablo ve nitelik adlarıyla ilgili, zorunlu olmayan ancak tüm geliştiriciler tarafından uyulan bazı kurallar bulunmaktadır. Öncelikle tablo adı, nitelik adı ve programlama içerisinde değişken adı, fonksiyon gibi sistematik bileşenlerin adlarında Türkçe karakter kullanılmaktan kaçınılmaktadır. Türkçe karakter için belirli karakter setlerinin kullanılması gerekliliği zaman zaman problemlere yol açtığı için artık bazı dillerde problem yaşanmasa da Türkçe karakter kullanmamak klasik bir yaklaşım haline geldi. Türkçe kelimelerden Türkçe karakterler kullanmadan faydalanılabileceği gibi doğrudan İngilizce karşılıklarını kullanmak da sık başvurulan bir yöntemdir.

Veri tabanlarında tabloların, programlamada ise sınıfların adları büyük harfle başlar ve eğer birden fazla kelimeden oluşuyorsa her kelimenin ilk harfi büyük seçilir.

Kullanicilar

SatisKayitlari

GecisBilgileri

uygun tablo adı seçimine birer örnek teşkil etmektedir.

ÖNEMLİ

Türkçe dilinin kullanımı konusunda hepimiz hassasiyetlere sahibiz. Bu sebeple burada yer alan yaklaşımlar eleştiriye sebep olabilir. Lütfen burada olayı teknik terim boyutuyla ele aldığımızı ve kabul gören yaklaşımları sunduğumuzu göz ardı etmeyelim. Bununla birlikte her ne kadar Türkçe ad kullanma yöntemini sunuyor olsak da gelecekte uluslararası projelerde de yer alabileceğinizi ve projenizi açık kaynak kodlu hale getirirseniz dünyanın her tarafından farklı insanların kodunuzu incelemek isteyebileceğini unutmayınız. Bu sebeple tüm anahtar kelime seçimlerinde programlamada ortak dil olan İngilizce kelimeleri seçmenizi tavsiye ederim.

Veri tabanları içerisindeki nitelik adları ve programlama dillerinde değişkenler ve fonksiyon adları küçük harfle başlar, birden fazla kelime içermesi durumunda her bir kelime büyük harfle başlar.

kullanici

erisimHakkiVarMi

sonGorulme

Eğer bir niteliği oluşturan kelimeler bir bütün değil de bir beraberliği ifade ediyorsa “_” işareti ile bağlanabilir. Örneğin bir kullanıcı ile o kullanıcıya ait ders bilgisi “kullanici_ders” şeklinde gösterilebilir. Bazı geliştiriciler birden fazla kelime içeren ifadeleri de “_” kullanarak belirtmektedirler. Örneğin “son_gorulme” de uygun bir kullanımdır.

Ham veri kümelerinin tek bir elektronik tablo yapısında bulunduğundan bahsetmiştik. Bu durumda satırlar ve sütunlar doğallıkla birbirleriyle bağlantılı oldukları için verinin bütünlüğünü tehdit eden herhangi bir unsur bulunmamaktadır. Ancak veri tabanı söz konusu olduğunda nitelikler gruplanmakta ve birden fazla tablo elde edilmektedir. Verinin birden fazla tabloya bölünmesi, normal şartlar altında veri bütünlüğünü tehlikeye sokmaktadır. Ancak veri tabanı yapısında tüm tablolar ve kayıtlar anahtarlar sayesinde veri bütünlüğünü korumaya devam etmektedirler. Anahtarlar, kayıtları temsil eden benzersiz değerler alan niteliklerdir. Birincil ve ikincil adı verilen iki anahtar türü bulunmaktadır.

Birincil anahtarlar (Primary Key, PK), bir tabloda her bir kaydı eşsiz olarak temsil eden değerleri içeren niteliğin özelliğidir. İkincil anahtarlar (Secondary/Foreign Key, FK) ise bir birincil anahtarın başka bir tablo içerisinde verileri birleştirme amacıyla yer alması durumunda sahip oldukları özelliktir. İlgili bölüm altında anahtarları ayrıntılı inceleyeceğiz ancak daha iyi kavramak adına bir örnek verebiliriz. Günlük hayatta sıkça kullandığımı kimlik numaralarımız birincil anahtardır. Kimlik numarası bir kişi hakkındaki tüm kayıtları temsil eden eşsiz bir değerdir (PK). Kimlik numarasının farklı bir tablo içerisinde bir kişiyi işaret etmek için kullanılması da mümkündür (FK).

3.2. Gösterim Şekilleri

Önceki başlıkta genel olarak kavramlardan bahsettik. Şimdi sistem tasarımı sürecinin ayrılmaz bir diğer bileşeni olan görsel gösterimleri ele alacağız. Elbette görsel gösterimler de tasarlanan sistemin belgelendirilmesi kadar önem taşımaktadır. Aynı zamanda yine teknik terimlerde olduğu gibi görsellerin de global geçerliliğe sahip olması önemlidir. Bir dokümandaki bir şema üzerinde yer alan en ufak bir işaretin, hem dokümanı hazırlayan, hem geliştiren, hem de gelecekte bu dokümanı inceleme potansiyeli olan ekiplerin aynı anlamı verecekleri bir yapı elde etmek gerekmektedir.

Sözü edilen motivasyon altında zaman içerisinde çeşitli gösterim şekilleri oluşturulmuştur. Bu gösterim şekillerinden sık bilinenleri; Chen, Kaz Ayağı (Crow's Foot) ya da Martin, Merise, IDEF1X ve EXPRESS'tir (Bachman, 1969; Gogolla, 1991; Menzel & Mayer, 1998, Purchase & diğ., 2004; Nizam, 2011). Hangi yöntem tercih edilirse edilsin, global bilinirliğe sahip bir yöntem tercih edildiğinde hazırlanan tasarım dokümanının geçerliliği ilgili yöntemin geçerliliğiyle orantılı olacaktır.

Bir tablo içerisinde yer alan nitelikler arasındaki fonksiyonel bağımlılıkları göstermek için de bir yöntemimiz bulunmaktadır. Fonksiyonel bağımlılık konusunu bir sonraki bölümde ayrıntılı inceleyeceğiz. Şimdilik hangi niteliklerin diğerlerini temsil edici özellikte olduğunu göstermemiz şeklinde özetleyebiliriz bu terimi. Fonksiyonel bağımlılıkta bir nitelik kümesinin başka bir nitelik kümesi tarafından temsili dokümante edilecektir. Notasyonun şablonu şu şekildedir:

$\{X_1,X_2\} \rightarrow \{X_3,X_4,X_5\}$

Anahtarlardan bahsederken kullandığımız kimlik numarası örneğini burada tekrar ele alalım:

$\text{kimlikNumarasi} \rightarrow \{\text{ad, soyad}\}$

Burada kimlik numarasının ad ve soyad için temsil niteliği taşıdığı bilgisini edinmekteyiz. Gösterimin sağında ya da solunda tek bir nitelik olması durumunda { … } gösterimini kullanma zorunluluğumuz bulunmamaktadır. Örnekte sol tarafta sadece kimlik numarası olduğu için doğrudan yazdık ancak sağ tarafta iki niteliğimiz olduğu için parantezli notasyonu kullandık. Bir varlığı metin olarak göstermek istersek aşağıdaki şablonu uygulayabiliriz:

Kullanici: kimlikNumarasi, ad, soyad

Bu notasyon kimlikNumarasi, ad ve soyad niteliklerinden oluşan Kullanici varlığını işaret etmektedir. Ayrıca kimlikNumarasi niteliğinin altı çizili olması, bu niteliğin temsil edici özelliğe sahip olduğunu vurgulamaktadır.

Bu ders kapsamında veri tabanı ile ilgili gösterimlerimizde Kaz Ayağı yönetimini kullanacağız. Bir sonraki alt başlıkta kaz ayağı yöntemini kullanarak çeşitli veri tabanı olgularını nasıl göstereceğimizi öğreneceğiz.

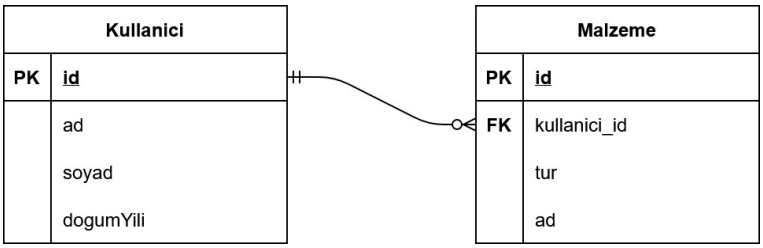
3.3. Kaz Ayağı

Dersin önceki bölümlerinde bir veri tabanı tablosunun nasıl gözüktüğüyle ilgili örnek vermiş ancak Kaz Ayağı gösteriminden bahsetmemiştik. Ders içerisinde daha önce kullandığımız gösterimler de Kaz Ayağı gösterimine uygundur.

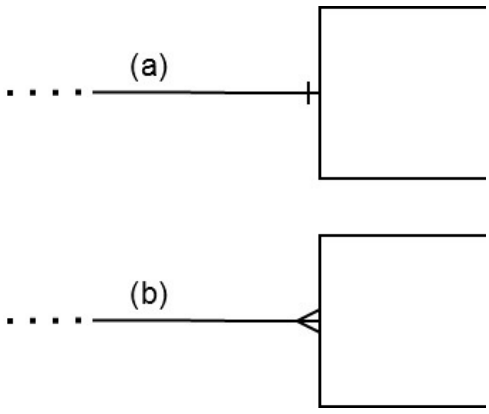
Şekilde, Kullanici adlı veri tabanı tablosunun gösterimi mevcuttur. Kutunun üst kısmında tablo adı, altında tabloda bulunan niteliklerin bir listesi, niteliklerin solunda ise varsa niteliklere ait anahtar özellikleri

belirtilmektedir. Anahtar özellikleri birincil anahtar için PK, ikincil anahtar için FK ifadeleriyle gösterilmektedir.

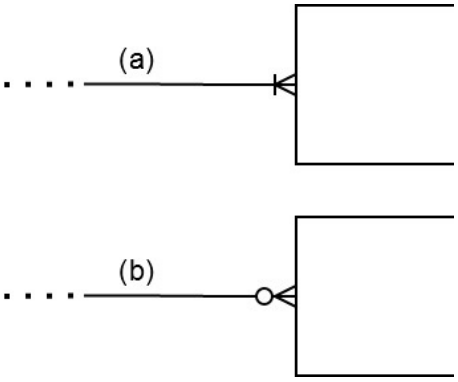
Birden fazla veri tabanı tablosunun aynı anda gösterimi için aşağıdaki örneği inceleyebiliriz.



Burada Kullanici ve Malzeme adında iki tablonun yer aldığını görmekteyiz. İki tablo için de nitelikler listelenmiştir. Kullanici tablosu için id niteliği birincil anahtarken, Malzeme tablosu için id birincil, kullanici_id ise ikincil anahtar olarak tanımlanmıştır. Ayrıca iki tabloyu birbirine bağlayan bir eğri olduğunu görmekteyiz. Bu işaret, iki tablonun birbirleriyle hangi nitelik üzerinden ve hangi türde ilişkili olduğunu göstermektedir. Görüldüğü üzere Kullanici ve Malzeme tabloları, Kullanici tablosu üzerinde id, Malzeme tablosu üzerinde ise kullanici_id nitelikleri üzerinden birbirleriyle bağlantılıdır. Bağlantının tablolara temas ettiği noktalarda bazı işaretlerin yer aldığını görmektesiniz. Şimdi bu işaretlerin anlamlarına odaklanalım.



Şekilde (a) gösterimi tek, (b) gösterimi ise çok bağlantı anlamındadır. Gösterim, kurulan bağlantıda, bağlanılan tabloda en fazla kaç kayıt ile bağlanılabildiğini göstermektedir. Örneğin Kisi ve Satis tablolarını düşünelim. Bu iki tablo arasındaki bağlantı, Kisi tablosuna tek, Satis tablosuna ise çok işaretiyle bağlanacaktır. Bunun sebebi, bir kişinin birden fazla satışta yer alma ihtimali olması ancak bir satışın yalnızca ve en fazla bir kişiyle bağlantılı olmasıdır. Bu sebeple Kisi'den Satis yönüne çok, Satis'tan Kisi yönüne tek bağlantı kurulur.



Bu örnekte ise (a) ve (b) ilişki gösterimlerinin ikisinin de çok türünde olduğunu görmektesiniz. Ancak iki gösterimin yanında da 1 ve 0'a benzeyen işaretler görüyorsunuz. Bu işaretler, bağlantı kurulan tabloda en az kaç kayıt bulunması gerektiğini göstermektedir. Yine Kisi ve Satis tablolarını ele alalım. Bir kişi hiçbir satış kaydıyla eşleşmeyebilir. Ancak bir satış kaydı mutlaka bir kişiyle eşleşmelidir. Bu sebeple bağlantının Kisi tablosuna bağlanan kısmı zorunluluğu ifade eden 1 gösterimiyle, Satis tablosuna bağlanan kısmı ise zorunluluk olmadığını ifade eden 0 gösterimiyle kullanılacaktır. Burada odaklandığımız konu gösterim

şeklidir. İlişki türleri ve nasıl tanımlanması gerektiğiyle ilgili konular gelecek bölümlerde tekrar ele alınacaktır.

Bölüm Özeti

Yeni bir sistem tasarlanması aşamasında tasarlanan yapının dokümante edilmesi oldukça önemlidir. Hem analiz ve geliştirme ekiplerinin "aynı dili" konuşabilmesi, bilgi aktarabilmesi, zaman içerisinde bilginin kaybolma riskinin ortadan kaldırılması gibi avantajlar hem de projenin gelecekte farklı ekipler tarafından devralınması durumunda herhangi bir bilgi kaybı yaşanmadan faaliyetlerin sürdürülebilmesi açısından dokümantasyon değerli bir süreçtir.

Veri tabanı tasarımında da bilgisayar ortamında uygulamaya geçmeden önce gerçekleştirilmesi gereken eylemler ve bunların dokümante edilmesi ihtiyacı bulunmaktadır. Oluşturulacak dokümanın global geçerliliğe sahip olması için bilinmesi ve uyulması gereken çeşitli kurallar bulunmaktadır. Bu bölüm içerisinde çeşitli genel terimler ve gösterim şekilleri ifade edilmiştir. Bu terimleri kısaca gözden geçirelim:

Veri kümesi: Bir veri parçasının elektronik tablo biçiminde satırlar ve sütunlar şeklinde gösterilmesidir. Genellikle MS Excel ve CSV formatlarında karşımıza çıkar.

Veri ambarı: Birden fazla veri kaynağından elde edilen verinin tek bir merkezde toplanarak, enformasyona dönüştürülmek üzere birlikte işlenmesidir.

Tablo: Veri tabanı içerisinde nitelikler birbirleriyle gruplandırılırlar. Bu sayede tek bir elektronik tablo biçimindeki veri kümesi, çok sayıda tabloya dönüşür. Veri tabanı için bu yapılara tablo (table) ya da varlık (entity) adı verilmektedir.

Sütun: Her bir tablo içerisinde yer alan dikey veri yapılarına sütun (column), nitelik (attribute) ya da değişken (variable) adlarının verildiği görülmektedir.

Kayıt: Tablolarda yer alan her bir satır için kayıt (tuple) kelimesi kullanılmaktadır.

Bilgisayar sistemleri açısından bir hataya sebep olmasa da tablo ve nitelik adlarının yazımında uyulan çeşitli kurallar bulunmaktadır. Aynı zamanda bu kurallar programlama da geçerlidir. Bir veri tabanı tablosu ya da programlamada bir sınıfın adı büyük harfle başlar. Eğer birden fazla kelime içeriyorsa her kelime büyük harfle başlar (BuBirTabloAdidir). Bir nitelik, programlama değişkeni ya da fonksiyon adı ise küçük harfle başlar ve birden fazla kelime için her kelime büyük harfle başlatılır (buBirNitelikAdidir).

Eğer bu adlarda farklı olgular ele alınıyorsa (kişi ve ders gibi) kelimeler küçük harfle başlayabilir ve "_" işaretiyle birbirlerinden ayrılabilirler (kisi_ders). Ayrıca seçilen adlarda Türkçe karakterlere yer verilmez.

Bir tabloyu metin olarak tarif etmek için aşağıdaki şablon kullanılabilir.

TabloAdi: anahtarNitelik, nitelik1, nitelik2, …

Bir tablo içerisinde yer alan niteliklerin birbirleriyle olan ilişkisini göstermek için fonksiyonel bağımlılık gösterimine ihtiyaç bulunmaktadır. Fonksiyonel bağımlılık, nitelikleri bir küme grubu olarak düşündüğümüzde, hangi kümenin diğerini temsil ettiğinin gösterilmesidir diye özetleyebiliriz. Gösterimi şu şekildedir:

$\{X_1,X_2\}\rightarrow\{X_3,X_4,X_5\}$

Bu gösterimde 1 ve 2 indisli nitelikler, 3, 4 ve 5 indisli nitelikler için temsil edici niteliktedirler.

Ders boyunca veri tabanı tasarımlarımızın gösteriminde Kaz Ayağı gösterim yönteminden faydalanacağız. Literatürde Kaz Ayağı (Crow's Foot / Martin)'nın yanında IDEF1X, Bachman, Barker, Merise ve EXPRESS yöntemlerinin de yer aldığını bilmekte fayda bulunmaktadır.

Kaynakça

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

Bachman, C. W. 1969. Data structure diagrams. ACM SIGMIS Database, 1(2), 4–10. https://doi.org/10.1145/1017466.1017467

Gogolla, M., & Hohenstein, U. 1991. Towards a semantic view of an extended entity-relationship model. ACM Transactions on Database Systems (TODS), 16(3), 369-416.

Menzel, C., & Mayer, R. J. 1998. The IDEF family of languages. In Handbook on architectures of information systems (pp. 209-241). Springer, Berlin, Heidelberg.

Nizam, A., 2011. Veritabanı Tasarımı: İlişkisel Veri Modeli ve Uygulamalar, Papatya Yayıncılık Eğitim.

Purchase, H. C., Welland, R., McGill, M., & Colpoys, L. 2004. Comprehension of diagram syntax: an empirical study of entity relationship notations. International Journal of Human-Computer Studies, 61(2), 187-203.