7. ÖZEL VERİ TABANI YAPILARI

Birlikte Düşünelim

Her zaman yapmamız gereken eldeki bir veri kümesini bir veri tabanına dönüştürmek midir?

İki tablo arasında başka türde ilişkiler tanımlanabilir mi?

Veri tabanında ne tür veriler saklanabilir?

Etrafınızda yer alan, en sık kullandığınız bilişim sistemlerini ele alın. Sizce arka planda nasıl bir veri tabanı yapısına sahip olabilirler? Hangi tablolar ve hangi ilişkiler kullanılıyor olabilir?

Verinin değişip değişmediğini anlamak için neler yapılabilir?

Bir veriyi saklamadan saklamak (sadece doğrulayabilmek) mümkün müdür?

Başlamadan Önce

Veri tabanları her zaman aynı türde tablolar, nitelikler ve ilişkiler içermezler. Bazı durumlarda, probleme ya da elde edilmek istenilen duruma göre bir veri tabanı tasarımı tasarlamak gerekebilir.

Bu bölüm içerisinde klasik yaklaşımdan farklı ancak yine de sık karşılaşılan özel durumlar ve bu durumlarda kullanılacak veri tabanı tasarımlarına yer verilmiştir. Bu bölümde öğreneceğiniz tasarımların birçoğunu doğrudan kullanacak, bazılarını da ihtiyaç duydukça daha büyük veri tabanlarının bir bileşeni olarak sürece dahil edeceksiniz.

Yıldız şema, hiyerarşik veri, özetleme, bağlantısız varlık ya da varlık grupları ve zaman damgasıyla ilgili süreçler ve örnekler, ilgili alt başlıklar içerisinde sunulmuştur.

7.1. Yıldız Şema

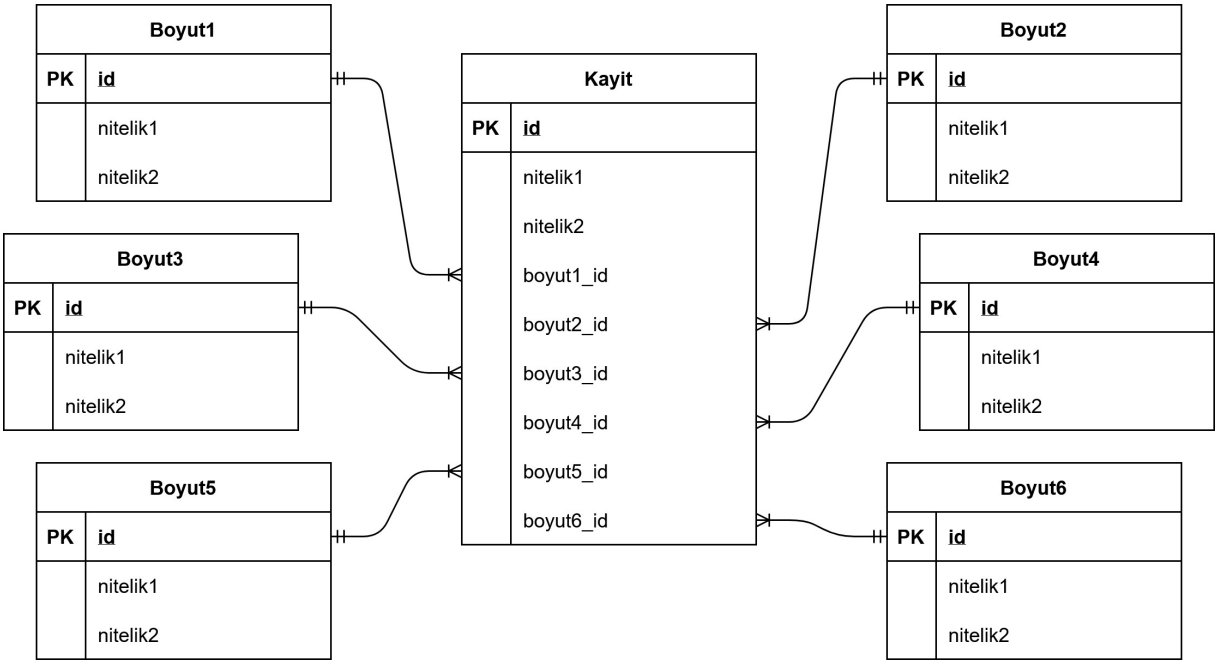
Sistemler, kullanım türlerine göre farklılıklar göstermektedirler. Veri tabanı içerisinde yer alan her bir tablo, genellikle bir olguyla eşleştirilir. Her bir bağlantı tablosu da bir eylemin kayda alınmasıdır. Dolayısıyla olguların yoğunlukla kayıt altına alındığı örneklerde çoğa çok bağlantı türüne sık rastlanmaz. Bir kütüphane örneğini alalım. Ancak bu örnekte kişilerin kütüphaneden kitap ödünç alma durumu yok. Yalnızca mevcut kitapların kayıt altına alınması isteniyor. Bu durumda kabaca bu veri tabanı yazar, kitap, dolap ve benzeri tablolar barındıracaktır. Her bir tablo, kütüphane içerisindeki olguları temsil edecek ve sınırlı sayıda kayıt alacaktır. Tablo adları birer olguyu işaret etmekle birlikte tablo içerisinde yer alan kayıtlar bu olguların miktarını gösterici niteliktedir. Şimdi bu örnekte kişilerin kitapları ödünç alabilmesi durumunu da ele alalım. Bu durumda tüm ödünç alma durumlarının kayıt altına alınması gerekecektir. Odunc (ödünç) adında bir tablo tanımlayalım ve ödünç alınan kitapların kayıtlarını bu tabloya eklediğimizi düşünelim. Bu tablo hangi tablolarla ilişkili olacaktır? Başka bir şekilde sorarsak bir ödünç alma eylemi hangi olgularla ilişkilidir? Bir "kişi", bir "kitap" ödünç alıyor değil mi? Dolayısıyla Kisi ve Kitap tablolarıyla ilişkili olduğunu rahatlıkla görebiliyoruz. Bununla birlikte az önce bir ipucu daha sundum. Ödünç alma "eylemi" diye vurguladım. Bu bir eylem, yani muhtemelen bir bağlantı varlığı olacak. Dolayısıyla burada çoğa çok bağlantı olduğundan bahsedebiliriz. Gerçekten de Kisi ve Kitap tabloları arasındaki bağlantıyı incelediğimizde çoğa çok bağlantı olduğunu görebiliriz.

Ele aldığımız kütüphane örneğinde kitap ödünç alınabilmesi ve alınamaması arasında bir fark bulunmaktadır. Odunc tablosu olması en büyük farklılık elbette ancak daha geniş bakarsak; ödünç alma eylemi bulunduğunda bu veri tabanına yapılan giriş sayısı çok daha artacaktır. Sadece mevcut kitapların kayıtlarının tutulduğu durumda kütüphaneye yeni bir kitap geldiğinde, bir kayıt güncelleneceği zaman ya da silinmesi gerektiğinde işlem gerçekleştirilecektir. Ancak ödünç alma durumu işin içine girdiğinde sayılan eylemlere ek

olarak her bir ödünç alma durumunun da kayıt altına alınması gerekecektir. Bu eylemin sıklığını da düşündüğümüzde belki de veri tabanındaki en hacimli ve en sık kullanılan tablonun bu olduğunu varsayabiliriz.

Bu türde, sıklıkla yapılan işlemlerin kayıtlarının saklandığı veri tabanı türüne “kayıtsal veri tabanı” (transactional database) adını vermekteyiz. Bu tür veri tabanlarında tüm işlemlerin yer aldığı, yüksek hacime sahip ve işlemlerin neredeyse tamamının üzerinde gerçekleştirildiği bir tablo bulunmaktadır. Otoban geçişleri, sınav sonuçları, etkinlik bileti satışı, banka hesapları gibi birçok durum bu kullanım şekline örnek olarak gösterilebilir.

Sıklıkla karşılaşılan bu veri saklama yöntemine bir ad verilmektedir. Yıldız şema (star schema) adı verilen bu tasarımda kayıtlar ve bu kayıtlarla ilgili olgular bir arada bulunmaktadır.



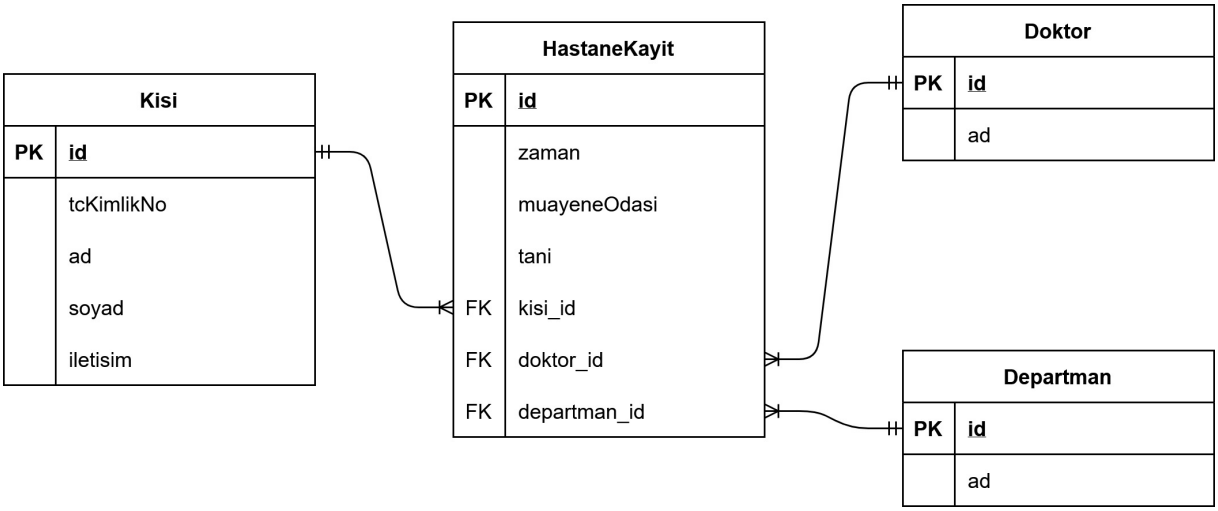
Şekilde bir yıldız şema örneği görmektesiniz. Ele alınan örnek her ne olursa olsun bu şema türü için tablolara verilen genel adlar bulunmaktadır. Elbette kendi veri tabanı tasarımınızda siz istediğiniz adı kullanabilirsiniz ancak bu adları da tabloların türü olarak aklınızda tutmanızı öneriyorum.

Yıldız şemada ortada yer alan tabloya kayıt (transaction) tablosu adı verilmektedir. Bu tablo az önce vurguladığımız tüm yoğun kayıtların tutulduğu, hacmi yüksek olan, veri tabanının kuruluş amacı olan kayıtları içeren tablodur. Bu tablo ve içerdiği kayıtlar elbette birçok olgu ile ilişkilidir. Örneğin bir otoban geçiş kaydında geçiş işlemleri bir kayıt tablosuna yazılacaktır elbette ancak bu kayıt araç, kişi, gişe, il, tarife ve benzeri birçok olgu ile ilişkili olarak kayıt altına alınmalıdır. Kayıtların ilişkili olduğu her bir olgu bir veri tabanı tablosu olarak gösterilir.

Şimdi öğrendiklerimizi birleştirelim. Kayıt tabloları eylemlere ait kayıtları içeren tablolardır. Eylemler genellikle bağlantı tabloları olarak karşımıza çıkar. Bağlantı tabloları olgular arasındaki çoğa çok bağlantı sayesinde ortaya çıkmaktadır. Bu durumda şemadaki veri tabanı tasarımında kayıt tablosu etrafında yer alan tüm tablolar çoğa çok bağlantılıdır ve bağlantı tablosu kayıt tablosunun kendisidir. Evet, bu doğru. Kayıt tablosunu bir nevi bağlantı tablosu olarak görebiliriz. Her ne kadar çoğa çok bağlantıdan doğduğunu düşünsek de bu durum bağlantı tablolarını küçümsemek için yeterli değildir. Bağlantı tabloları genellikle veri tabanı içerisinde en çok kaydı içeren; veri bütünlüğünü sağlamak için en çok ihtiyaç duyulan tablo türüdür. Yıldız şema da bunu oldukça net olarak göstermektedir.

Yıldız şema tek başına kullanılabileceği gibi bir veri tabanı yapısının bir bileşeni de olabilir. Eldeki veri tabanı tasarımı içerisinde birçok tablo olabilir. Ayrıca kayıtsal eylemler için bağlantı tabloları da bulunabilir. Bu durumda boyut adı verilen, olguları bulunduran tablolar da olacaktır ancak veri tabanında kayıt tablosu dışında yer alan tüm tabloların bu özellikte olmasına gerek yoktur. Bir e-ticaret sistemini ele alalım. Satis (satış) tablosu kayıt tablosuna güzel bir örnektir. Gerçekleştirilen her bir satış işlemi bu tabloya kaydedilecektir. Bir eylemin kaydı için de kullanıldığı aşikardır. Olgulara göz gezdirdiğimizde, bir satış için

kişi ve ürün olgularına ihtiyaç duyduğumuzu söyleyebiliriz. Genellikle bu tarz sistemler çok daha karmaşık olurlar. Dolayısıyla çok daha fazla boyut tablosuyla ilişki sağlanacaktır. Ama tüm tablolar değil! E-ticaret sistemi veri tabanında satışlarla doğrudan ilişkili olmayan çeşitli kayıtlar bulunması da mümkündür. Bu durumda bu tarz bir veri tabanı tasarımında yıldız şema yalnızca bu tasarımın bir bileşeni olabilir.



Şemadaki örnekte bir hastanedeki kayıtları görüyoruz. Ortada yer alan HastaneKayit tablosunu Muayene tablosu olarak da düşünebiliriz. Bu sefer boyut tablolarını inceleyerek başlayalım. Kisi tablosu hastaneye gelen herkesin bir kez kaydedildiği tabloyu ifade etmektedir. Hastaneyi ziyaret eden toplam kişi sayısı kadar kayda sahip olması beklenecektir. Bir olguyu ifade eder. Herhangi bir eylem barındırmamaktadır. Doktor tablosun da kişilerin kayıt altına alındığı bir tablodur. Ancak bu tabloya kaydedilen kişiler hastanede çalışan doktorlardır. Hastanede çalışmış olan toplam doktor sayısı kadar kayda sahip olması beklenir. Departman tablosu da hastanede yer alan bölümlerin bilgilerini barındıran tablodur. Gördüğünüz gibi tüm boyut tabloları olguları kayıt altına alan tablolardır. Bu tablolara yeni kayıt eklenmesi ya da değişiklik yapılması oldukça sınırlı olacaktır. HastaneKayit tablosu ise hastaneye yapılan tüm girişlerin kayıt altına alındığı tablodur. En çok kaydın ve en çok işlem yapılan tablonun bu olması beklenecektir.

Burada şu ayrıntı da dikkat çekebilir. Bu veri tabanı tasarımının işlevi bir MS Excel dosyası ile de sağlanabilir. Nihayetinde bir kişinin, bir departmanda görev alan doktorla, belirli bir zaman ve mekanda muayene eylemini gerçekleştirmesi kayıt altına alınmaktadır. Bu bilgi geleneksel elektronik tablo ile de kayıt altına alınabilir. Ancak şimdi kayıt tablosunun açık halini göz önüne getirin. Her bir satırda kişinin bilgileri, doktorun bilgileri, departman adı ve gerekli diğer bilgilerin tekrar etmesi gerekecektir. Eğer bu veri geleneksel elektronik tablolar ile yönetilirse aynı verinin çok fazla kez tekrar edilmesi kaçınılmazdır. Ancak bu yapıyla birlikte her bir olgu farklı bir tablo ile kayıt altına alınmakta; kayıt tablosunda yalnızca bu olguların id'leri kullanılmaktadır. Böylece tekrar eden şey yalnızca bir tam sayı olacak, en performanslı şekilde tüm veri, bütünlüğü bozulmamış biçimde saklanabilecektir.

Bu ayrıntıyı şu şekilde de ele alabiliriz: Bir potansiyel kayıt tablosu ele alalım. Bu tablo içerisinde sıkça tekrar etmesi olası nitelikleri belirlemek zor olmayacaktır. Bu nitelikleri birer kategorik değişken olarak düşünebiliriz. Yani bir değer kümesi içerisinde yer alan değerleri yüksek frekansla tekrarlı şekilde alan nitelik kümeleri belirleyebiliriz. Bu durumda ortada yalnızca bir kayıt tablosu varken boyut tablolarını da belirlemek ve veriyi göz önünde bulundurarak veri tabanı tasarımını elde etmek zor olmayacaktır.

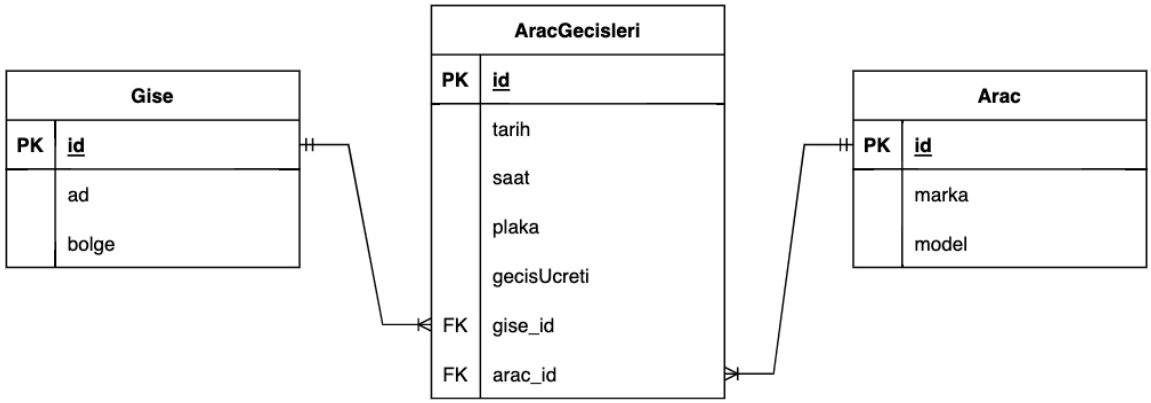
Aşağıdaki tabloyu ele alalım:

AracGecisleri: tarih, saat, gişe, bolge, plaka, marka, model, gecisUcreti

Burada AracGecisleri adında bir tablo ve bu tablo içerisinde yer alan nitelikleri görüyoruz. Sürekli yeni kayıtlar alması muhtemelen bir tabloyla karşı karşıyayız. Bunu bir yıldız şema olarak değerlendirme kararı vererek kayıtları içerecek tabloyu merkeze alarak üzerinde çalışmaya başlayabiliriz.

Tüm geçişleri kayıt altına alacak tablomuzun adı yine AracGecisleri olsun. Boyut tablolarını belirlemek için bu tablo içerisinde tekrar eden nitelikleri belirlemek yeterli olacaktır. tarih ve saat niteliklerini kayıt tablosunda bırakmak en mantıklısı olacaktır. Bir gişeden çok sayıda geçiş yapılabileceği için bu tablo içerisinde aynı gişe bilgisini çok kez görebiliriz. Bu sebeple gise niteliğini bir boyut tablosu olarak

değerlendirebiliriz. Aynı zamanda bolge niteliği de sıkça tekrar edecektir. Bununla birlikte bolge ve gise niteliklerinin bağımlılık barındırdığını da söyleyebiliriz. Bir gişe bir bölgeye bağlı olmak zorundadır. Dolayısıyla bu iki niteliği tek bir boyut tablosu olarak birlikte değerlendirebiliriz. plaka niteliği zaten birincil anahtar olarak kullanılabilecek; aracı ifade eden bir nitelik olduğu için kayıt tablosunda kalabilir. Araçla ilgili birden fazla nitelik olsaydı bir boyut tablosu tanımlamak daha akıllıca olacaktı. Aynı olguyu ifade eden birden fazla nitelik olması durumunda bu nitelikleri gruplayarak bir boyut tablosu oluşturmak ve bu boyut tablosunu kayıt tablosunda temsil edebilecek bir nitelik belirlemek en iyi çözümdür. marka ve model nitelikleri aracın özelliklerini belirten iki niteliktir. Bu iki niteliği gruplayarak bir boyut daha elde edebiliriz. Bir arac_id niteliği ile kayıt tablosunda temsil edilmesi mümkündür. Burada marka ve model niteliklerinin neden plaka niteliğiyle gruplanmadığını düşünmüş olabilirsiniz. Bir aracın plakası zaman içerisinde özellikle satış işlemlerinde değişebilmektedir. Bu sebeple bir araç ile plaka her zaman birlikte anılmayabilir. Bu örnekte bu risk göz önünde bulundurulmuş ve ilgili nitelikler birlikte gruplandırılmamıştır. gecisUcreti niteliği sayısal bir niteliktir ve her geçiş için farklı değer alabilir. Ayrıca zaman içerisinde de güncellenen bir değer olacağı için bir boyut olarak ele alınması uygun olmayacaktır. Tüm bu yorumlar ışığında aşağıdaki çözümü doğru kabul edebiliriz.



Şekilde AracGecisleri adında bir kayıt tablosu ile Gise ve Arac adlarında iki boyut tablosu görülmektedir. Böylece küçük çaplı da olsa bir yıldız şema veri tabanı tasarımı elde etmiş olmaktayız.

7.2. Hiyerarşik Veri

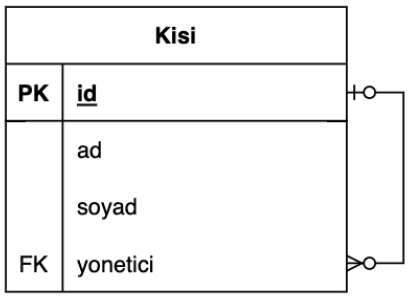
Veri tabanı yapısı içerisinde her bir tablonun genellikle bir olguyu, bağlantı tablolarının ise eylemleri kayıt altına aldığından bahsetmiştik. Olgular birbirleriyle çeşitli şekillerde bağlantı kurabiliyorlar. Bu bağlantı türü çoğa çok olduğunda bir bağlantı tablosunu da tasarımımızın bir parçası yapmaktayız.

Veri tabanı içerisinde kurulan bağlantılar genellikle iki olgu yani dolayısıyla iki tablo arasındaki kurulmaktadır. Ancak aynı olgu da kendisiyle bir ilişkiye sahip olabilir. Olguyu programlamadaki sınıf (class) kavramına, her bir kaydı da bir nesneye (object, instance) benzeterek bu durumu daha iyi açıklayabiliriz. Kisi (kişi) tablosunu ele alalım. Bu tablo kişilerin kayıt altına alındığı, genel olarak bir kişi kaydının hangi niteliklere sahip olması gerektiği bilgisini barındıran olgudur. Bu tablo içerisinde yer alan her kayıt, kişi olgusunun birer örneğine işaret eder. Kisi, genel olarak bir kişiyi ifade ederken bu tabloda yer alan kayıtlar gerçek kişilerdir. Tablolar arasındaki ilişkiler tanımlanırken olgular arasında kavramsal olarak sonuç elde edilir. Ancak kayıtlar da işin içine girdiğinde gerçek olaylar kullanılmış olacaktır.

Bir kişi, herhangi başka bir olguyla etkileşimde olabilir. Bir ürünü satın alabilir, öğrencilik kaydı oluşturabilir, otel rezervasyonu yaptırabilir, bir şirkette çalışabilir. Peki bir kişi başka bir kişiyle bir etkileşimde olabilir mi? Fiziksel dünyada bunu düşünmek oldukça kolay. Elbette bir insan başka bir insanla herhangi bir şekilde etkileşimde olabilir. Bir kişi başka bir kişinin yöneticisi olabilir, ebeveyni olabilir, vârisi olabilir. Şimdi veri tabanı açısından düşünelim. Bir Kisi tablosu içerisinde yer alan iki kişi kaydını ele alalım. Bu kayıtlardan biri diğerinin yöneticisi olsun. Bu bilgiyi bu tabloda nasıl saklayabiliriz?

Veri tabanımızı en atomik şekilde kurguladık. Elimizde yalnızca bir Kisi tablosu var ve içerisinde sadece 2 tane kayıt var. Bu iki kayıt birbiriyle ilişkili. Farklı tablolar arasındaki ilişkileri belirlemeye oldukça alışığız

ancak aynı tablo içerisindeki ilişkileri tanımlamakla ilgili bir örnek görmedik. Yanıtı bulmak şimdiye kadar öğrendiklerinizle mümkün, ancak biraz zor olabilir. Şemayı inceleyelim.

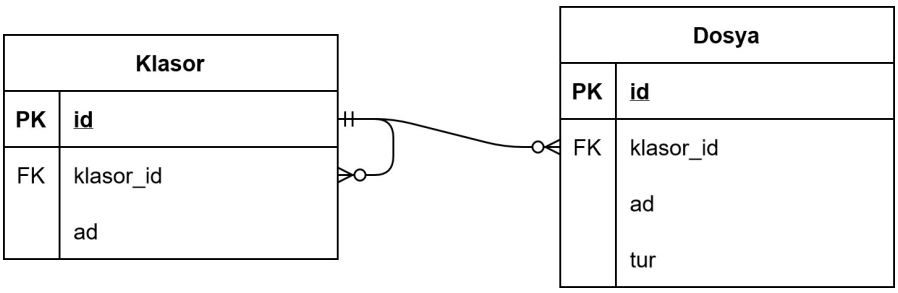


Lütfen okumaya devam etmeden önce tabloyu ayrıntılı olarak inceleyin. Klasik bir veri tabanı ilişkisine çok benzer bir yapıyı tek bir tabloda görüyoruz. Tablo, ilişki, birincil ve ikincil anahtarlar, nitelikler üzerinden kurulan bağlantı mevcut ancak tek bir tablo var. Standart yaklaşımımızı uygulayabiliriz. Kisi tablosunda yer alan bir kayıt, Kisi tablosunda birden fazla kayıt ile eşleşebilir mi? Yönetici olma durumunu göz önünde bulundurduğumuzu unutmayın lütfen. Yanıtımız evet. Bir kişi birden fazla kişinin yöneticisi olabilir, dolayısıyla birden fazla kayıt ile eşleşebilir. Şimdi ilişkinin diğer yönünü inceleyelim. Kisi tablosunda yer alan bir kayıt, Kisi tablosunda birden fazla kayıt ile eşleşebilir mi? Bu soru öncekiyle aynı gibi gözükse de ilişkinin tersini irdelediğimizi unutmayın. Bu sefer kişinin yöneticisini araştırıyoruz. Yanıtımız hayır. Bir kişi yalnızca tek bir yöneticiye sahip olabilir. Yani özet olarak; bir kişi birden fazla kişiye yöneticilik yapabilir ancak bir kişinin en fazla bir yöneticisi olabilir. Bu durumda çoğa bir (ya da bire çok) ilişki ile karşı karşıyayız. Bu örneği daha net anlamak için tablomuzu açık açık inceleyelim.

id	ad	soyad	yonetici
1	Emre	Akadal	
2	Lorem	Ipsum	1
3	Dolor Sit	Amet	1

Tabloda üç kayıt yer alıyor. Birinci satır için yönetici niteliği değer almamış. İkinci ve üçüncü satırlar yonetici olarak 1 değerini almışlar. Bu durum; ikinci ve üçüncü kayıtların yöneticisinin 1 id'li kaydolduğunu göstermektedir. Bu durumda id'si 1 olan kayıt, 2 tane kaydın yöneticiliğini yapmaktadır. Ancak id'si 2 ve 3 olan kayıtlar ancak tek bir yöneticiye sahip olabilmektedirler. Tabloda yönetici olan kayıt, birden fazla kayıtla eşleşebilirken, diğer kayıtlar en fazla tek bir kayıtla eşleşebilmektedirler.

Farklı bir örnek ele alalım. Bir bulut dosya saklama sistemi için veri tabanı tasarımı gerçekleştiriyor olalım. Yine en temel bileşenlerini göz önünde bulunduracağız. Klasörler, dosyalar ve bunların veri tabanına yerleşimini anlamaya çalışalım. Klasör ve dosyalar farklı birer olgu kabul edilebilir ve iki ayrı tablo olarak tutulabilir. Adları Klasor ve Dosya olsun. Bir Klasör birden fazla dosya içerebilir ancak bir dosya ancak tek bir klasör içerisinde yer alabilir. İki tablo arasında bire çok ilişkiden söz edebiliriz. Ancak burada bir ilişki daha bulunmalı. Klasörler de kendileri arasında bir hiyerarşiye sahipler. Bir klasör başka bir klasörün içinde yer alabilir ya da bir klasör birçok klasör ya da dosyayı içeriyor olabilir.



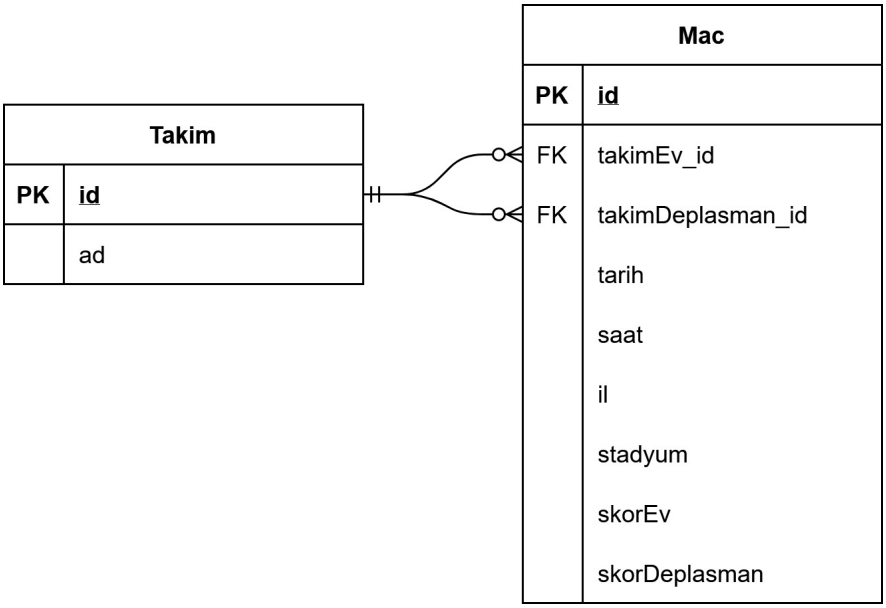
Buradaki ilişki türünü incelediğinizde, bir klasör birçok klasör içerebilir ancak bir klasör ancak tek bir klasörün içinde yer alabilir yorumu sayesinde bire çok ilişki ile devam etmemiz gerektiğini fark edeceksiniz. Böylece klasörler, bir klasor_id sayesinde hangi klasör altında yer aldıkları bilgisini saklamaktadırlar. Ele alınan bir klasör için, klasörün id'si kullanılarak klasor_id'si bu klasörün id'sine eşit kayıtlar aranılabilir,

böylece seçilen klasör içerisindeki diğer klasörler listelenebilir. Aynı sorgulama Dosya tablosunda da yapıldığında klasör altındaki dosyalar listelenebilir.

Bu yaklaşım bir hiyerarşi içeren her türlü yapılanma için kullanılabilir. Şirketlerdeki insan kaynağı yapılanmaları, yönetim teşkilatları, kamu kurumları ve bölümleri, askeriye, üniversitelerde yer alan birimler bu tarz bir veri tabanı tasarımı kullanılarak ifade edilebilirler.

Şimdiye kadar ele aldığımız örnekler, bir tablonun kendisiyle bire çok ilişkili olduğu örneklerdi. Bildiğiniz üzere bire çok ilişkide iki tablo doğrudan bağlanır ve ilişki tipi çok olan yönde ikincil anahtar yerleştirilir. Ancak çoğa çok bağlantılı tablolarda bir bağlantı tablosu sağlanmalıdır. Bir tablo kendisiyle çoğa çok bağlantılı olduğunda tablo yapısı nasıl kurulmalı? Bir tablo hangi durumda kendisiyle çoğa çok bağlantılı olur? Yine örnek üzerinden gidelim.

Futbol takımları bir tablo içerisinde listelenebilirler. İki futbol takımının birbiriyle en bariz etkileşime geçtiği eylem nedir? Maç yapmak olsa gerek. İki futbol takımı maç yapma eylemi ile kayıt oluşturabilir. Ayrıca bu eylemi çok kez gerçekleştirebilir. Bir takım çok sayıda takımla maç yapabilirken, bir takım yine çok sayıda takımla maç yapabilir. Yanlış gibi gözükse de bu cümle ilişki tipini iki açıdan da incelememizi sağlıyor. Burada çoğa çok bağlantıdan bahsettiğimizi söyleyebiliriz.



İncelediğiniz örnekte Takim ve Mac adında iki tablo bulunmaktadır. İki tablo arasında iki bağlantı var gibi görülebilir ancak gerçekte Takim tablosu kendisiyle çoğa çok bağlantılıdır. Mac tablosu ise bu çoğa çok bağlantıdan kaynaklanan bağlantı tablosudur. İki takımın gerçekleştirdiği maç eylemi bu tablo içerisinde kayda alınır. Ve evet, bu bir yıldız şema örneğidir.

7.3. Özet Fonksiyonları

Bilgi sistemlerinin en önemli tarafı, herhangi bir güvenlik zafiyeti oluşturmadan bir bilgiyi saklayabilmektir. Günümüz şartlarında da bilgi zafiyeti oluşturulması en muhtemelen iki veri türü; kişisel veriler ve erişim bilgileridir. Kişisel verilerin korunmasıyla ilgili çeşitli yasal düzenlemeler bulunmaktadır. Bununla birlikte kişisel verinin türü değişken olduğu için teknik boyutta alınabilecek kesin kararlar bulunmamaktadır. Ancak erişim bilgileri konusunda atılabilecek, özellikle veri tabanı boyutunu ilgilendiren konular mevcuttur.

Erişim bilgilerinden kastımız bir sistem ya da özelliğe erişirken kullandığımız parola, anahtar kelime ve benzeri özel ifadelerdir. Her ne kadar veri tabanına erişimi kısıtlıyor olsak da internet ortamına açık sistemler, yazılımda, işletim sisteminde, kullanılan kütüphanelerde ve diğer bazı bileşenlerde meydana gelebilecek açıklar sebebiyle izinsiz kullanıcıların erişimine açılabiliyor. Bir veri tabanının tamamını görüntüleme yetkisi, eğer kullanıcılara ait erişim bilgilerini de içeriyorsa tüm kullanıcıların zarar görmesine sebep olabilmektedir. Bununla birlikte bazı kullanıcıların aynı şifreyi farklı sistemler için kullanması, şifrenin herhangi bir sistemde ele geçirilmesi sonrasında başka sistemlerde de zafiyet oluşmasına sebep olabilmektedir. Bu riski aşmak için sıklıkla kullanılan yöntem şifreyi veri tabanına doğrudan saklamamaktır.

Şifre yerine şifrenin doğrulanmasını sağlayacak bir içeriği saklamak, şifrenin ele geçirilmesine engel olmaktadır. Bu da bizi özet fonksiyonları ile tanıştırıyor.

Özet (hash), bir girdi ifadesi sonucunda sabit uzunlukta bir çıktı üreten fonksiyona verilen addır (Ajao & diğ., 2019; Todorova & diğ., 2019). Birkaç örnek inceleyerek devam edelim.

Değer	Özet
1	c4ca4238a0b923820dcc509a6f75849b
Emre	34a1988c0a5878ddcea0cdb42d651e1e
İstanbul Üniversitesi Açık ve Uzaktan Eğitim Fakültesi Bilgisayar Programcılığı Diploma Programı Veri Tabanı Tasarımı Dersi	d8df3d089c195a8d3d66475a6ea51df0

Tabloda yer alan ilk sütun çeşitli değerler içerirken, ikinci sütun ise bu değerlerin MD5 fonksiyonuyla üretilmiş özet değerlerini içermektedir. İlk değer yalnızca 1 karakterlik bir ifadeyken özeti 32 karakterlik bir değerdir. İkinci satır 4 karakterlik, üçüncü satır çok daha uzun bir metni içermektedir. İkinci ve üçüncü satırdaki değerlerin özetleri de 32 karakterlik özet değerlere sahiptir. Özet fonksiyonları, aldıkları her bir girdi değer için bir çıktı üretirler. Girdi değer birebir aynı olduğunda ürettikleri değer de aynı olacaktır. Girdi değerlerde en ufak bir değişiklik bile özetin tamamen değişmesine sebep olmaktadır.

Veri tabanında saklanılması gereken, içeriği önemsiz ancak doğrulanma ihtiyacı duyulan içerikler özetlenerek saklanabilirler. Bir kullanıcının şifresinin içeriğinin ne olduğu önemsizdir. Ancak kullanıcı sisteme tekrar eriştiğinde şifresini doğru girip girmediği kontrol edilebilmelidir. Bu sebeple kaydolma sürecinde şifre özetlenerek veri tabanına kaydedilir. Kullanıcı sisteme tekrar eriştiğinde giriş yapmak için şifresini girer, şifre yazılım tarafından tekrar özetlenir ve elde edilen özetle veri tabanına kaydedilen özetin aynı olup olmadığı karşılaştırılır. Özetler aynı ise kullanıcı sisteme kabul edilir.

Özet fonksiyonları tek yönlü çalışan fonksiyonlardır. Bir girdi değer alarak bu girdi değere karşılık özet alabilirken; bir özet alarak bu özeti oluşturan değeri elde etmek teorik olarak mümkün değildir. Pratikte ise şöyle bir yaklaşım izlenebilir: Mümkün olan tüm değerler için özetleme yapılarak değer-özet ikilisinden oluşan bir kütüphane elde edilebilir. Böylece eldeki özet bu kütüphanede araştırılarak eğer kaydedildiyse hangi değer kullanılarak oluşturuldu öğrenilebilir. Buna “Brute Force” saldırısı adı verilmektedir. Aşmak için kombine de edilebilen iki yöntem kullanılır. Bunlardan birincisi şifrenin belirli uzunlukta, sayı, harf ve özel karakter içeren metinlerden oluşturulmasıdır. Bu kombinasyona sahip tüm olasılıkların bir kütüphanesinin hazırlanması neredeyse imkansız olduğu için erişim zorlaştırılacaktır. Bununla birlikte sistem için özel bir anahtar belirlenir ve tüm özetlemelerde bu anahtar da özetlenecek değerle birleştirilir. Böylece özetlenecek değerin çok basit olması durumunda dahi anahtar olarak kullanılan kısmın bilinmiyor olması sebebiyle özetin tersine döndürülmesi mümkün olmayacaktır.

Özellikle internet tabanlı sistemlerin saldırılara maruz kalma imkanının yüksek olması sebebiyle bu tür güvenlik önlemlerinin alınması oldukça önemli ve gereklidir. Özetleme sayesinde alınacak önlemler, bazı sızmalar olması durumunda dahi şifrelerin ele geçirilmesini engelleyecektir. Ancak bu yöntem kişisel verilerin korunması konusunda fayda sağlayamayacaktır.

7.4. Bağlantısız Varlıklar

Dersin bu bölümüne kadar ilişki niteliklerin tabloları oluşturmasından, ilişkili tabloların da veri tabanlarını oluşturmasından bahsettik. Peki veri tabanında yer alan tablolar hiçbir tablo ile ilişkili olmayabilir mi?

Zaman zaman veri tabanında yer alan ancak diğer tablolarla doğrudan bağlantılı olmayan tablolar ya da tablo grupları bulunmaktadır. Eğer tablolar, doğrudan bir ilişki tipi ile bağlı olmayabilirler. Bir tablo, doğrudan hiçbir tabloyla bağlantılı olmayabilir. Ya da iki bağlantılı tablo grubu, birbirleriyle bağlantılı olmayabilirler.

SistemAyarlari	
PK	id
	anahtar
	deger

Görselde, SistemAyarlari adında, sistem dahilinde sürekli kontrol edilmesi gereken ve bileşenlerin genel hareketlerini yönlendiren ayarların bulunduğu bir tablo gösterilmektedir. “id” adında bir birincil anahtar kullanılmıştır. Bununla birlikte tüm ayarlar “anahtar” ve “deger" nitelikleri kullanılarak tabloda sağlanmıştır. Bu tablo, doğası gereği diğer tablolarla ilişkili değildir. Ancak yönetilen sistem için sistem ayarlarının saklanılması, ihtiyaç halinde güncellenmesi ve gerektiği anda hızlıca çağrılması ihtiyacı sebebiyle veri tabanı içerisinde bulundurulmaktadır. Dolayısıyla bu tablonun diğer tabloların hiçbiriyle bağlantılı olmaması bir hata değildir.

Veri tabanı tasarımı içerisinde bir tablo diğer tablolarla bağlantısız olabileceği gibi bir tablo grubu da bağlantısız olabilir. Birbirleriyle bağlantılı olan bir tablo grubu, yine birbirleriyle bağlantılı olan başka bir tablo grubuyla hiçbir tablo üzerinden bağlantı kurmuyor olabilir. Bu durum genellikle birden fazla bileşene sahip sistemlerde görülür. Bir firmaya ait bilgi sistemi içerisinde ürün, müşteri, nakliye, fiyatlandırma, bayi yönetimi, insan kaynakları ve benzeri çok sayıda bileşen yer alabilir. Bu bileşenler genellikle kendi içerisinde birkaç tablodan oluşur ve diğer bileşenlerle ilişkili olabilir ya da olmayabilir.

Eğer bir analiz gerçekleştirmek için ikincil veriye başvurduysak yine bağlantısız tablolardan oluşan bir veri tabanı elde edebiliriz. Gerçekleştirilecek bir analiz için toplu taşıma kullanım verilerini elde ettiğimizi düşünelim. Sonrasında bu veri ile hava durumu arasındaki ilişkiyi anlamak için hava durumu kayıtlarını da elde edelim. Tüm bu kayıtları tek bir veri tabanına yerleştirdiğimizde, toplu taşıma kullanımıyla ilgili verilerle hava durumu verileri arasında tablolar arası bir ilişki olmadığını fark edeceksiniz. Buradaki ilişki, analiz sürecinde araştırmacının tarih ve varsa bölge bilgilerini eşleştirmesiyle elde edilecektir. Bölge bilgileri üzerinden tablolar arası ilişki kurulması mümkün olsa da iki veri kaynağında da bölge belirtme şekilleri birebir aynı olmalıdır. Aksi halde bir ön işleme süreci de işletilmelidir.

7.5. Zaman Damgası

Zaman kontrolü veri tabanında sıklıkla yapılan işlemlerden birisidir. Bir kaydın ne zaman eklendiği, en son ne zaman güncellendiği, geçerlilik süresinin ne olduğu gibi birçok zaman bilgisi ihtiyacı bulunmaktadır. Bu ihtiyaçlar elbette ki bir nitelik içerisinde tarih ve saat bulundurmakla çözülebilir gibi gözüküyor. Ancak bu tür nitelikler metin içerikli olacağı için, zamanlar üzerinde bir işlem yapılmaya çalışıldığında birçok zorluk da beraberinde gelecektir. En basit uygulama olarak bir tarihin geçip geçmediğini anlamak, tarih ve saatlerin karşılaştırılabilir olmasını gerektirir. Ancak bunu doğrudan yapmak kolay olmayacaktır. Ya da bir zaman bilgisi üzerine 1 gün eklemek gibi bir işlem yapmak oldukça zor olacaktır.

Bu gibi eylemler sıklıkla ihtiyaç duyulan eylemler olduğu için elbette bir çözüm yöntemi de üretilmiş durumdadır. Zaman damgalarının veri tabanına kaydedilmesiyle ilgili iki tür zaman formatı bulunmaktadır. Bunlardan birincisi ISO 8601 ile tanımlanmış format (27), diğeri ise Unix Timestamp olarak bilinen zaman damgasıdır (11).

ISO 8601; tarih, saat ve zaman dilimi bilgilerini özel bir biçimde sunan bir yöntemdir. Yöntem, alfanümerik içeriğe sahip olsa da farklı sistemler tarafından aynı şekilde yorumlanması ve üzerinde kıyaslama ya da hesaplama yapabilecek yazılımlar olması sebebiyle tercih edilebilmektedir. Bir örneği şu şekildedir:

2022-09-16T11:45:30+03:00

Örnek şu bilgiyi içermektedir: 2022 yılı, 9. Ay, 16. gün, saat 11:45’in 30. saniyesi. Zaman dilimi ise +3 olarak belirlenmiş.

İkinci yöntem ise GMT zaman diliminde 1 Ocak 1970 saat 00:00’dan beri geçen toplam saniye sayısını veren Unix Timestamp’tir. Bir örneğini inceleyelim.

1663318419

Bu zaman damgası türünün dezavantajı, bakıldığında doğrudan anlamlandırılamamasıdır. Bu sayının hangi tarihe denk geldiğini anlamak için bile bir dizi matematiksel işlem yapmak gerekmektedir. Ancak bununla birlikte çeşitli avantajları bulunmaktadır. Birincisi sayısal veri olduğu için saklaması ve üzerinde işlem yapması kolaydır. Bu şekilde kaydedilmiş bir zaman damgasının 1 gün sonrasını hesaplamak için 86400 (1 günlük saniye sayısı) eklemek yeterlidir. Ya da zamanın geçip geçmediğini anlamak için doğrudan matematiksek büyüktür ve küçüktür işaretleri kullanılabilir.

İki zaman damgasının da kullanılıyor olmasının yanında Unix Timestamp türünün daha yaygın ve kullanışlı olduğunu söylemek gerekir. Özellikle oyunlarda bir eylemin gerçekleştirildiği ya da gerçekleştirileceği zaman üzerine yapılacak işlemler için bu türdeki zaman damgaları oldukça kullanışlıdır.

Zaman damgası hesaplaması manuel olarak gerçekleştirilmez. Neredeyse tüm programlama dillerinin zaman damgalarıyla ilgili fonksiyonları bulunmaktadır. Bu sebeple bir zaman damgasının üzerinde nasıl işlem yapılacağına dair kafa yormaya genellikle ihtiyaç duyulmamaktadır. Zaman damgası hesaplaması konusunda otomatik hesaplamalar yapan; tarihi zaman damgasına, zaman damgasını ilgili zamana çeviren web sayfaları da bulunmaktadır.

Bölüm Özeti

Veri tabanı tablolarını olgusal ve eylemsel olarak ikiye ayırabiliriz. Bir veri tabanı oluştururken ortaya çıkardığımız ilk tablo türü olgusal tablolardır. Kullanici, Arac, Kitap, Bolge ve Kategori bu türde tablolara örneklerdir. İki tablo birbiriyle çoğa çok bağlantılıysa bir bağlantı tablosu elde edilir. Bu tablo eylemsel olarak bahsedebileceğimiz tablolardır. Satis, Gecis ve TakipEtme eylemsel tablolara örnektir.

Yıldız şema, bir bağlantı tablosunun ortada, bu bağlantı tablosuna bağlantılı olgusal tabloların çevresinde olduğu yapıdır. Ortadaki tabloya kayıt, etrafındaki tablolara boyut tablosu adı verilmektedir. Ortada yer alan tablo genellikle çok sayıda kayıt alır. Boyut tabloları ise sabit içerikleri saklayarak kayıt tablosunun bunları id ile temsil etmesini mümkün kıldığı için veri hacminde büyük azalma sağlar. Aynı zamanda veri tekrarının da önüne geçilmiş olur. Bu veri tabanı türü başlı başına bir veri tabanı yapısı olabileceği gibi bir veri tabanı yapısının bir parçası da olabilir.

Tablolar her zaman birbirleriyle ilişkili olmak zorunda değildir. Bir tablo kendisi üzerinde bir ilişkiye de sahip olabilir. Bunun sebebi tabloların olgusal olması ve aynı türe ait iki olgunun bir eyleme sahip olabilmesidir. Örneğin Kullanici tablosunu ele alalım. Herhangi bir sosyal ağ üzerinde kullanıcıların birbirini takip etmesi sıklıkla görülen bir özelliktir. Bu özellik, Kullanici tablosunun kendisiyle ilişkili olması ile sağlanabilir. Burada ilişki tipi çoğa çok olacağı için bir bağlantı tablosu bulunması gerekecektir. Bu da TakipEtme tablosu olabilir. Böylece bir Kullanici, bir de TakipEtme tablosu elde edilecektir. TakipEtme tablosu üzerinde iki tane ikincil anahtar olması gerekmektedir. Hangi kullanıcının hangisini takip ettiği bilgisi bu şekilde sağlanacaktır. Zaten bağlantı tablolarında en az iki tane ikincil anahtar bulunduğunu daha önce öğrenmiştik. Bir tablo kendisiyle bire çok bağlantılı da olabilirdi. Bu durumda tablo üzerinde bir ikincil anahtar bulunması gerekir. Bu türde bağlantılar genelde hiyerarşiyi gösteren bağlantılardır. Örneğin bir dosya saklama sisteminde klasörler ve dosyaların hiyerarşisi bu türde bir veri tabanı yapısıyla saklanabilir. Her Klasor bir klasor_id sayesinde üst klasörünü işaret edecektir.

Özet fonksiyonları değişken uzunluktaki bir girdiye karşılık her zaman aynı uzunlukta ve aynı çıktıyı üreten tek yönlü fonksiyonlardır. Bir özet fonksiyonu için 1 karakterlik bir girdi de sağlanabilir; GB'lar boyutunda bir dosya da sağlanabilir. Her türlü girdi için eşit uzunlukta bir çıktı üretilecektir. Çıktının uzunluğu kullanılan özet fonksiyonuna göre değişir. Aynı girdi her zaman aynı çıktıyı üretirken girdideki en ufak değişiklik çıktıyı tamamen değiştirir. Özetleme tek yönlüdür. Bir girdi için özet üretilebilir ancak özet kullanılarak girdi elde edilemez. Bu fonksiyonlar genelde veri tabanı içerisinde şifre gibi içeriği önemsiz ancak doğrulanabilirliği önemli değerlerin saklanılmasında kullanılır. Bir şifre, özetlenerek veri tabanında saklanır. Tekrar kullanılması gerektiğinde sisteme girilen şifre tekrar özetlenir ve veri tabanındaki özet ile karşılaştırılır. Böylece şifrenin kendisi kaydedilmezken doğrulanabilmesi mümkün hale gelir.

Veri tabanı içerisinde bir tablo ya da tablo grubu, veri tabanının geri kalan kısmıyla bağlantılı olmayabilir. SistemAyarlari gibi yalnızca anahtar ve deger niteliklerinden oluşan ve ayarları saklayan bir tablo, veri tabanında yer alan hiçbir tabloyla bağlantılı olmayacaktır. Eğer eldeki sistem birçok bileşenden meydana geliyorsa bileşenler arasında bağlantı olmaması da mümkündür. İşletmeler genellikle çok sayıda bileşeni tek veri tabanında toplarlar. Tüm bileşenler firma için önemli olsa da veri tabanı açısından bağlantısız olabilirler. Bu bir hata değildir. Benzer şekilde bir analiz gerçekleştirilirken de farklı kaynaklardan alınan ikincil veriler doğrudan ilişkili olmayabilir ancak birlikte analiz edilmek üzere aynı veri tabanı içerisinde bir araya getirilebilirler.

Kaynakça

Ajao, L. A., Agajo, J., Adedokun, E. A., & Karngong, L. 2019. Crypto hash algorithm-based blockchain tech- nology for managing decentralized ledger database in oil and gas in- dustry. J—Multidisciplinary Scientific Journal, 2(3):300–325.

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

Todorova, M., Stoyanov, B., Szczypiorski, K., Graniszewski, W., & Kor-dov, K. 2019. Bentsign: keyed hash algorithm based on bent boolean function and chaotic attractor. Bulletin of the Polish Academy of Sciences: Technical Sciences, pages 557–569.